G.LC. BRAYAN MELRONI A5043742
M450 ECA

MYT 450 ECA

A Prototype for Human Resource Employee Management System for a medium-scale company

TABLE OF CONTENTS

-

1. Problem Description

1.1 Nature and context of the problem

New Age Private Limited established in Kandy Sri Lanka is a well-known Organization for manufacturing computer hard disk drives. Even though the company's manufacturing plant is situated in Kandy Sri Lanka, New Age has four sales offices located within Sri Lanka itself in Bambalapitiya, Nugegoda, Kurunagala & Kandy respectively.

The manufacturing plant consists of three departments: manufacturing department, human resource department and accounts department. A sales office also consists of three departments: sales department, accounts department & human resource department. According to current records available in all human resource departments altogether 1500 people are employed.

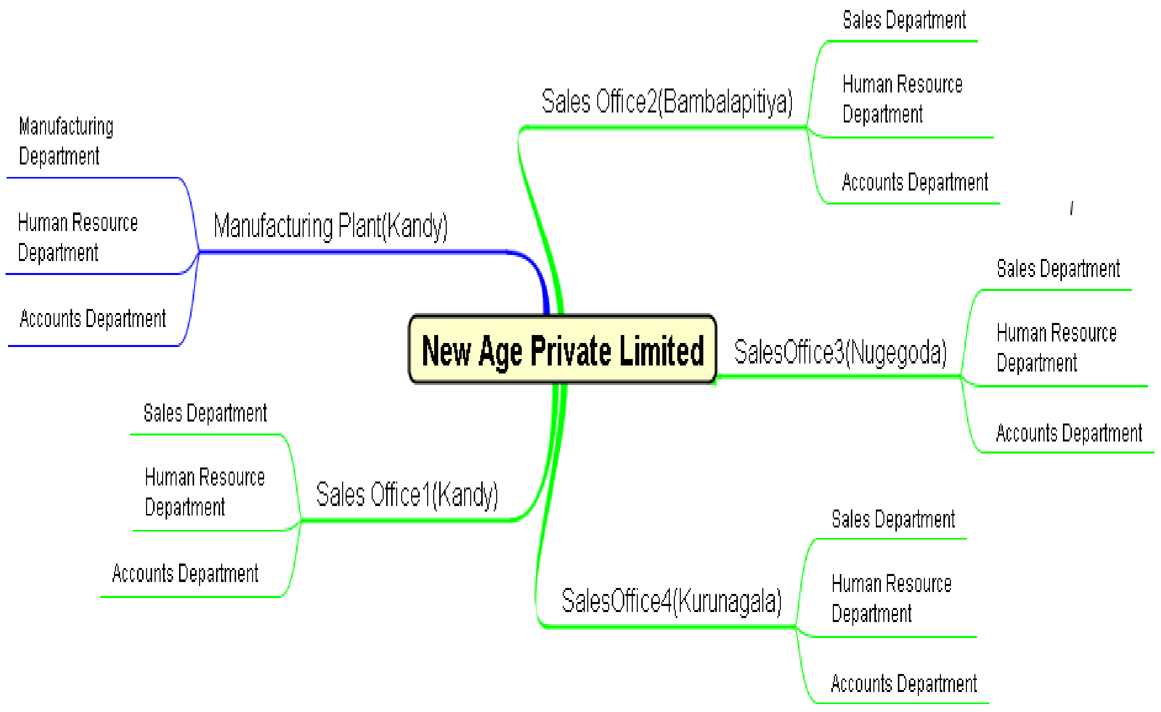
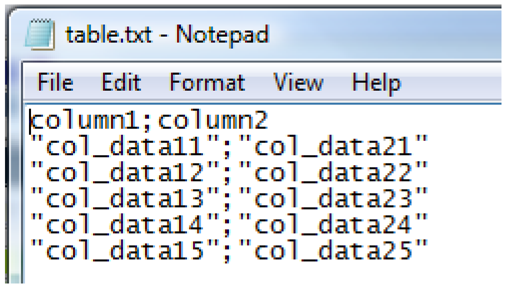


Figure 1: Structure Of The Organization

Each human resource department has a human resource manager. Other departments often need to know details of the employees who fulfill certain criteria. Some times a department in a particular place needs to know information about employees satisfying a certain criteria in other location. The most frequent criteria are employees coming from a particular place, employees who were working more than a particular time period, employees having a salary within a particular limit & employees having certain skills. It 's the human resource manager's duty to supply required information about employees. Often other departments like to have employee information in a form of well-organized format. Further more human resource department itself needs to find out employee information very often.

In four places employee information is stored as flat files in local computers. Those flat files are in .txt format. In two places commas delimit the files. In other two places semicolons delimit files. In all four places the first lines of the flat files had column names. I am giving below a sample flat file just to demonstrate the situation. Please consider that this is not even part of the actual existing flat files. Since I have signed an agreement with New Age Corporations that I am not giving their data to third parties I can't give exact files here.



```
column1;column2
"col_data11";"col_data21"
"col_data12";"col_data22"
"col_data13";"col_data23"
"col_data14";"col_data24"
"col_data15";"col_data25"
```

Figure2: Sample flat file

When employee information is needed human resource managers go through the files and create a report using Microsoft Word. In the other place employee information is stored in a MS Access 97 database & when information is needed human resource manager queries the database & creates a report using Microsoft Word. Currently the reports created in different places are in different formats.

In current setting when a department needs the information about an employee who are from other place the human resource manager in the place where information is required gives a telephone call & inform about that to the human resource manager in the place where the employees exist. Then the human resource manager in the place where employees exist create the required report & fax the report to the requested human resource manager.

Most frequently the human resource department in Kandy wants to know information about employees having certain skills within all places. In current setting Kandy human resource manager collects reports from all places & creates a report.

Further more it is found that same data exists in multiple files where data is stored in flat files. It was clearly seen because in some reports same employee information was mentioned repeatedly.

Other things is in current setting when the employee wants to know how his/her personal details are stored in the human resources department he/she has to fill a form & request for his/her details. Then human resource manager go through the data & create a report about the requester's information & hands it over to the requester.

-

When the detail of a particular employee changes or if an employee thinks that his/her data is incorrect & it should be changed he/she doesn't have a proper way to inform it to the human resource department. In current setting the employee has to meet the human resource manager & inform it.

The problem here is some human resource managers can't understand the reports generated in other places. Report generation itself has become a very much tiring experience for the managers. Sorting employee data satisfying a certain criteria is time consuming. There's no efficient mechanism for searching within all places. It has been found that some generated reports contain errors as well. Employees also feel that finding out their personal information is time consuming & inefficient. Further more they feel if some data about themselves change informing it to the departments also time consuming. Apart from that currently if anyone can get access to the local machine where flat files are stored he/she can get access to the confidential employee data automatically.

It is considered as a problem mainly due to below mentioned facts.

- Due to the fact that there doesn't exist a standard format for the reports some human resource managers can't understand the reports created at other places. That in turn makes the data inconsistent. Apart from that there's a different issue with reports as well. In the reports currently generating at the place where Microsoft Access 97 used, if the employee is not a trainer also everywhere it's mentioned as not trainee. The manufacturing department heads think if the person is not a trainee that information is not needed.

- When one human resource manager needs a report from another place giving telephone calls, waiting, receiving it as a fax message is time consuming.

- When a report is needed about employee satisfying a certain criteria from all places, collecting five reports and aggregating a report is time consuming, tiring as well as inefficient.

- Since data is stored in flat files in some places when creating reports filtering data, sorting data becomes very much tiring. This has resulted lot of errors in the reports. Further more data maintenance also hard when flat files are used.

- Since data exists in multiple places disk spaces are wasted.

- Enough security measures are not applied to confidential employee details especially for the flat files. In current settings flat files can be opened without authenticating.

- Employees believe that there should be an efficient quick way to see their employee details. Further more they find it very tiring to inform a change to the human resource department. They don't have time to meet the human resource managers and inform the changes. They need a quicker way to do this.

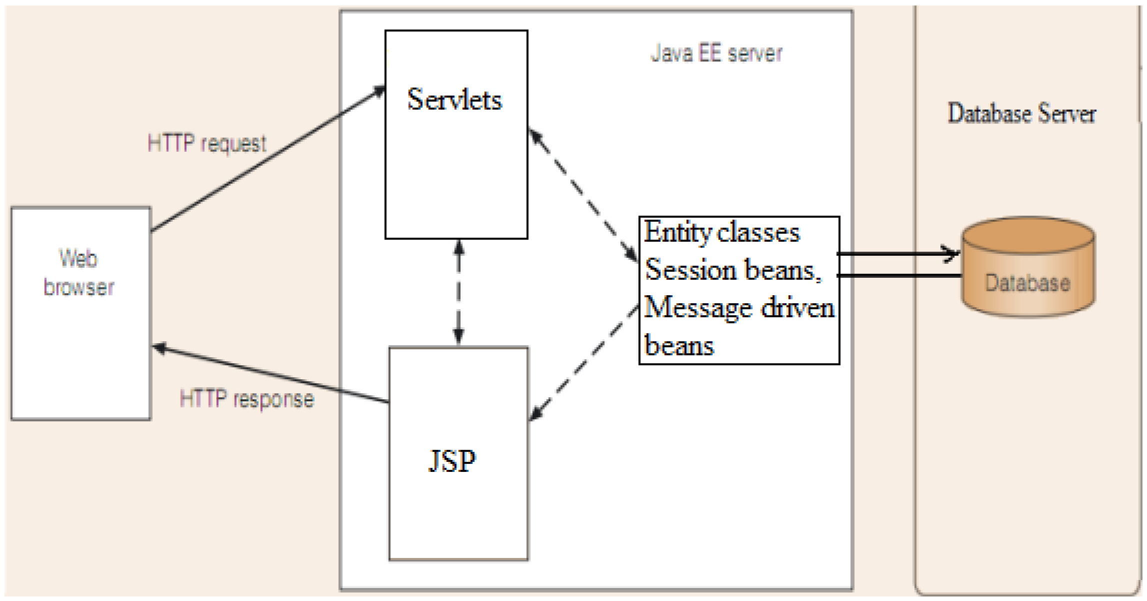
That's the problem I am trying to solve here.

1.2 Proposed solution

The solution I propose to give for the above-mentioned problem is a Human Resource Employee Management System with smooth data handling & generating reports in a unique format while maintaining data security as well. It will satisfy the above-mentioned employee needs as well as human resource managers' needs. Anyhow my solution is devoted to provide a software prototype that means it can be further improved.

The proposed system will be developed in an intranet & users will be able to access it using an Internet browser by typing the required URL. I am developing the server side components & will be deploying those in a J2EE compatible server. Actually I will be using Sun Java System Application Server Platform Edition 9.0_01.

I am giving here a technical overview of my proposed solution here. One by one an introduction of the server side components I am going to develop is presented below.



<u>Figure 2:The technical overview of the proposed solution</u>

<u>Reference</u>
The Open University (2008) M362 *Developing concurrent distributed systems,* Unit 8, 'The web tier ', Antilles, NIIT.

<u>1.Database at Database Server</u>

The first thing I propose is to have a centralized database where all employee information of five places is stored at one particular location. I think that place should be the Human Resource Department at the sales office Kandy since most of the time heads at that place need filtered employee information satisfying a certain criteria considering whole employees.

-

Here I plan to use the existing database structure it self. I will be using "Data Loader Version 3.6" software from Interface Computers to convert existing MS Access database into a MySQL database as well as to convert flat data files into MySQL database files. I download that software from http://www.dbload.com/download.htm
After that I will be using "MySQL Delete (Remove) Duplicate Entries 7.0" to remove the duplicate entries in the database if exists. I have downloaded that software from http://mysql-delete--remove--duplicate-entries.smartcode.com/info.html

2.Entity classes at Java EE Server

Then I plan to map all existing tables into entity classes. Data retrieving, modifying, deleting, entering new data will be done with the help of entity classes.

3.Session beans at Java EE Server

After that I will code all the business methods (methods for sorting, filtering, retrieving, modifying, deleting data, creating the reports) in session beans. I implement security to those methods using annotations. So that each method will be accessible only to users allowed in annotations.

4. Message Driven beans at Java EE Server

If the employee finds out some error in his/her records the ability to inform errors to human resource managers is provided through a message driven bean. Here I am using a queue based messaging that means point-to-point messaging model. I will develop it accordingly in such a way that when an employee sends an error message it is saved in the database. Human resource manager can view errors when required.

5.Servelt classes

Then I will code the controlling classes Servlets. Only the servlets can call the methods declared in the session beans as well as sending messages to a Queue. Further more servlets will be handling the most user input that goes to session bean methods as input. There I am using a combination of declarative security & programmatic security for the autherisation.

6.Java Server Pages

While developing servlets I will concurrently code the Java Server Pages for the graphical manifestation of the data as well. That's what is visible to the end user. Most of the time my Java Server pages will be dynamically generated according to user requests**. When required JSP pages will call the servlet methods. Servlet methods will call the session bean methods, do the processing & forward the output to a JSP page.**

Authentication of the users will be doing using a form-based authentication at very first before the users are allowed to go further. I am getting help of a database realm here.

Technical details of my solution will be further explained in the latter part. Now I am giving two diagrams to further clarify the system in the perspective of user's interaction.

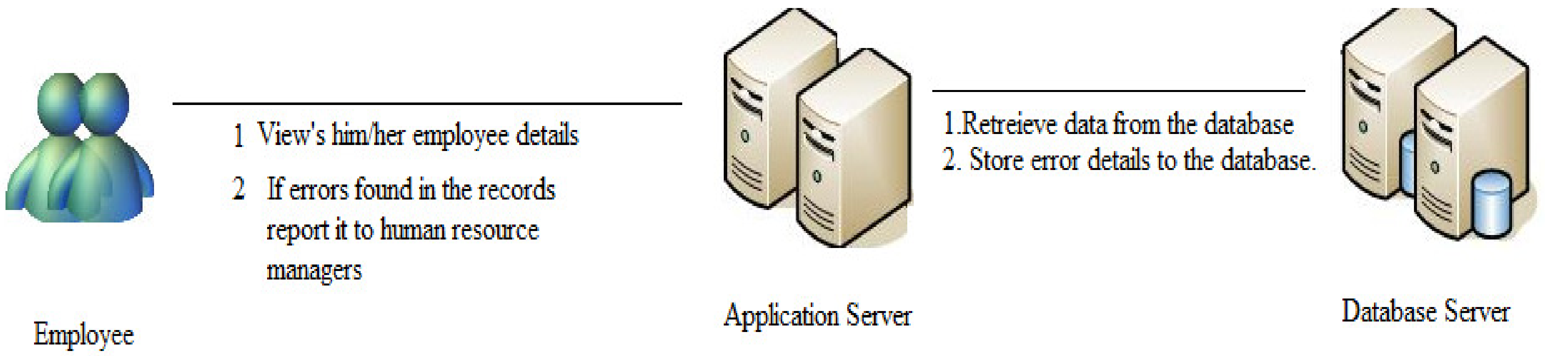

1 View's him/her employee details
2 If errors found in the records report it to human resource managers
1.Retreieve data from the database
2. Store error details to the database.
Employee
Application Server
Database Server

Figure 3: Employee's Interaction with the solution

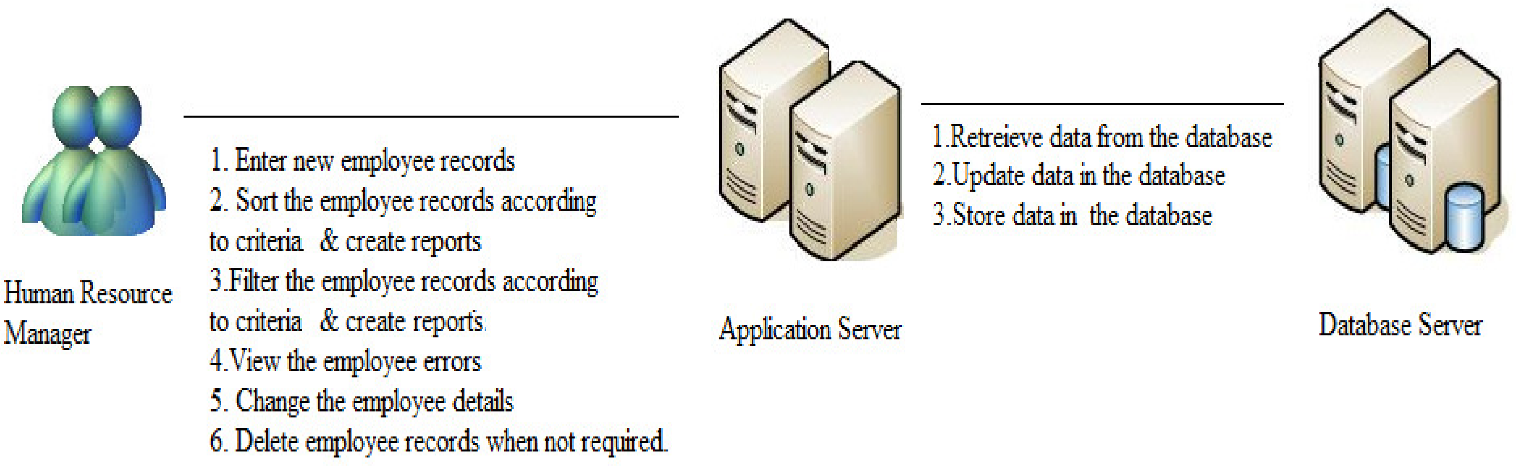

1. Enter new employee records
2. Sort the employee records according to criteria & create reports
3.Filter the employee records according to criteria & create reports.
4.View the employee errors
5. Change the employee details
6. Delete employee records when not required.
1.Retreieve data from the database
2.Update data in the database
3.Store data in the database
Human Resource Manager
Application Server
Database Server

Figure 4: Human Resource Manager's Interaction with the Solution

-

1.3 Analysis of likely impact

As a result of my proposed solution the employees as well as human resource managers will get a lot of impacts to the current settings. Both the parties will have to use a computer in the intranet when using the proposed system. Even though this is a hardware manufacturing company there are a very few amount of employees who are not very much familiar with software applications. So a small training also should be provided. It won't be a big issue since the numbers of employees need training is less.

Further more the company should employ a database administrator & an application server administrator to have a smooth operation. Company agreed with me that it's ok with them. Some times one person may be able to perform both.

Human Resource Managers

- Creating reports of the employees satisfying a certain criteria becomes efficient & easy for them. They will not go through text files & type the reports in Microsoft Word. They will not query Microsoft Access databases. In spite of that they will just fill up the criteria they want to filter and press a button. The report will be auto generated.
- When a human resource manager at a particular place needs to know employee information satisfying a certain criteria at another place he/she doesn't have to call to that particular place & ask the human resource manager at that place for it. No faxing. Just filling the criteria in a form & pressing a button. It's done.
- When employee information about the whole employee that satisfying a certain criteria is needed no collecting five reports & aggregating one report is done. Managers can create the required report by just one or two clicks.
- The reports will be easily understandable since all of the reports comply with a certain well-organized format. So that Other managers, heads will really understand the reports easily. They will like the reports.
- The data will not be stored in multiple places & disk space will not get wasted. The reports are guaranteed to be error free.
- Maintaining employee information will be easy since the flat files are not used.
- Employee data will be secure than the current settings.

Employees

- When the employee wants to know his/her personal details stored in the human resources department they will not fill paper forms and wait until their requests are processed. They will just use a computer in the intranet and log in to the system & view their details. They will find it easy & efficient as well as effective.

- When the details of a particular employee change he/she will not have to meet the human resource manager & inform it. They will log in to the site & type the errors & press “send”. They will find it very much relaxing, easy & less time consuming.

Ethics & ethical behavior

Here I am listing the stakeholders involved, their legitimate interests & the way I evaluate their legitimate interests.

Stakeholder:	Employees
Legitimate Interest:	To find out their employee information easily & efficiently. If employee information changes or he/she finds that there’s some error in details it should be easy to inform it to human resource managers.
Evaluation:	Employees find it’s easy to view their information & easy to report the alterations to the human resource manager. Since one employee can view only his/her information & can’t view other’s information there doesn’t exist ethical problems here.

Stakeholder:	Human Resource Managers
Legitimate Interest:	have an efficient, easy to use, effective employee report generation facility. To have all the generated reports comply with a well-organized easily understandable format To have a smooth flowing employee information management system To allow the employee to inform about changes of his/her information as soon as possible
Evaluation:	The human resource mangers can generate required employee reports easily & effectively. The reports are understandable to all heads of the departments. Human Resource managers will get informed about employee information changes quickly. The possibility of human resource managers giving the employee confidential information to a third party is prevented by an already signed agreement with the human resource department & the employee, that department is not giving the employee information to third parties. So there won’t be any ethical problems.

Stakeholder:	Heads of the other departments
Legitimate Interest:	have a well-organized easily understandable report containing the information of employees satisfying a certain criteria

-

Evaluation:	The reports are generated in a well-organized easily understandable way & they take management decisions by going through the reports easily. Since they have also signed an agreement with the company that they are not giving employee information to third parties there won't be any problems. So no ethical problems will occur.
Stakeholder:	Database Administrator/application Server administrator
Legitimate Interest:	have the server side components working correctly & maintaining the database efficiently.
Evaluation:	They will make sure the system functions properly & database is managed properly. He/She also has to sign an agreement with the company that he/she is not giving employee information to third parties. Then no ethical problems will occur.
Stakeholder:	Me
Legitimate Interest:	giving a prototype solution to the existing problem at New Age Corporations human resource departments
Evaluation:	I have also signed an agreement with the company that I am not giving the employee information to third parties so that no ethical problems will occur. I will give a working/workable prototype solution to the existing problem.

<u>Projects Involving human participants</u>

Since my project involves human participants I have gone through these questions.

Q1. **Does the proposed project involve collecting data or information from or about human participants?**
Yes- Information has to be collected from employees as well as human resource managers.

Q2. **Does the project involve working with National Health Service (NHS) patients or staff?**
No, it doesn't

Q3. **Does the project involve working with children or young people?**
No

Q4. **Would any reasonable judgment conclude that no harm (physical, psychological, etc.) could possibly rise to any person, living or dead, in connection with your proposed project?**

No it wouldn't

Here proposed solution is implemented as an internal employee management system & only employees & human resource managers will use the proposed solution. So that developing proposed solution won't affect the Open University UK or any other organization's repute or status.
My last decision is there doesn't seem to be a necessity for an ethical clearance for my work from Open University UK.

2.Account of Related Literature

Here I have grouped each Literature under the main issue, which it is discussing about. One by one I will account below.

2.1 Converting Microsoft Access database into MySQL database:

When converting Microsoft Access database in to MySQL database I wanted to find out the procedure it's done. So that I searched the official web site of the Data Loader software & found a step-by-step procedure. Site itself gives the screen shots of the way it is done. I am mentioning below the procedure as it was on the site.

Reference:
Data Loader (2009) *Data Loader Tour* [Online]. Available from: http://www.dbload.com/tour.htm [Accessed: 10 August 2009]

1. Starting a new session
2. Selecting the source & destination database types (In this case source type is MS Access & Destination type is MySQL)
3. Selecting the source database & the tables that need to be converted to MySQL.
4. Selecting the target MySQL database
   4.1: Entering username, password, server name, port number & clicking connect button. Then a list of databases in the server will be populated.
   4:2: Selecting the database that needed to be loaded.
5. If needed changing the destination table names (I am not changing the table names in my solution).
   Further more columns can be filtered using where conditions if needed. (I am not filtering in my solution. I am using it as it was in the original database)

6. Selecting the time that conversion need to be performed. There are three options "At once & Schedule it", "Run it now only and don't save", "Save and Schedule it to run at future time."
   I am selecting "Run it now only and don't save" option.

7. Clicking the start button & doing the conversion.

I have gone through exactly the above-mentioned procedure & already converted existing Microsoft Access databases to MySQL databases. Flat files also I converted using the Data Loader software according to the above-mentioned procedure. The difference was I had to select the format of the text files, delimited format in my case commas in two files & semicolons in two files. Further more I had to select check box saying first row has column names.

This information was very much helpful for forwarding me within the project since if I had to enter the data by hand it would have been a lot of pain. So that productivity of work was increased.

2.2 Defining Entity Classes:

I read a chapter of a book regarding entity classes to have some understanding about entity classes. Full reference of the Chapter is mentioned below.

Sriganesh, R.P., Brose, G. & Silverman, M., (2006). Java Persistence: Programming with entities. *In Mastering Enterprise JavaBeans 3.0*. Indianapolis: Wiley. p. 129-155.

After reading that Chapter I summarized main concepts that is helpful for my proposed solution.

- Each & every table in the database maps to an entity class. Each & every column name of the table will be mapped to attributes of the entity class. An object of the entity class corresponds to a row in the database.

- All the variables declared should be declared as private & for each variable there exists a public getter method as well as a public setter method.

- All the entity classes should have public no arguments constructor. Then public constructors with arguments should follow it.

- It's better to implement the entity class from java.io.Serializable interface. If so instances of the entity classes can be passed by reference to the clients for their local manipulation.

- The mechanism to do mapping entity classes with the relational schema is provided by the Java Persistence framework with the help of annotations.

- Annotations are defined in the **`javax.persistence`** package. So that first of all it's easy if whole package is imported otherwise one by one each annotation also can be imported. Then the class should be annotated with **`@Entity`** annotation to specify that it is an entity class.

- The instance variable that is analogous to the primary key should be annotated with the annotation **`@Id`**.

- When managing entity classes there will be a need of a thing called "persistence unit". Persistence unit can be thought as a logical grouping of entity classes, mapping metadata, & database configuration data. Entity classes have to be packed & deployed in persistence unit, which is defined as a descriptor file **`persistence.xml`**.

-

Book gives a simplest possible persistence.xml file & a description of other possible attributes & their child elements. So that after reading I created a possible prototype of a persistence.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0"
xmlns=http://java.sun.com/xml/ns/persistence ">
      <persistence-unit name="ems-PU" transaction-type="JTA">
      <provider>Provider's implementation of SPI  Class </provider>
            <jta-data-source>jdbc/employee</jta-data-source>
            <properties>
                  <property name="prop-name " value=" some-value"/>
            </properties>
      </persistence-unit>
</persistence>
```

`<Provider>`	full class name of the persistence provider's implementation of the SPI class
`<transaction-type>`	Attribute Of `<persistence-unit>`. Possible values of this element are `JTA` or `RESOURCE_LOCAL`.
`<jta-data-source>`	to mention the JNDI name of the data source that is being used.
`<properties>`	Vendor specific configuration properties

- When finding entities queries can be used. In situations where single query is used within the whole persistence unit or where changing the query is needed then named queries can be used. The following is an example for a named Query taken from the book.

```
@NamedQuery(name="findThem", queryString="SELECT a FROM Account a")
```

The above book gave me a real understanding about the underlying technology in Entity classes. I got understanding about how to structure an entity class, how the coding for an entity class should be organized, How to use annotations, why we have to define a persistence unit, how it has to be defined. So this chapter helps me to develop my solution further.

2.3 Further more On Entity Classes:

Then I wanted to have a look at more examples on entity classes. So I had a look on some pages from an OU (M362) course book.

Reference:
M362 *Developing concurrent distributed systems* (2008) Unit 7, The Open University. p. 26-35

Apart from the details I got to know from the previous resource regarding entity classes I learnt following facts by reading the above pages.

- The class is annotated with annotation **`@Table(name= "table_name")`** to specify the table name mapped with the class. If the table name & the class name is the same (when case differences are neglected) then this annotation is optional. It should be immediately followed by **`@Entity`** annotation.
- When considering instance variables if they differ from the corresponding column names neglecting the case difference, variables should be annotated using **`@Column(name="col_name")`** annotation.
- It is better to override the **`toString()`** method to display the primary key So that if exceptions are thrown by run time infrastructure it becomes meaningful.

This information further more sharpened my knowledge about entity classes. This OU book explains the information in a much more simpler way than the previous book. However now I am in a very good position to understand, implement entity classes due to both of the books.

In both books I couldn't find an example for a Named Query that need to be changed frequently. So I searched java API for further reference.

Java EE API (2007) *javax.persistence Annotation Type NamedQuery* [Online].
Available from:
http://java.sun.com/javaee/5/docs/api/javax/persistence/NamedQuery.html
[Accessed: 15 August 2009]
I have mentioned below what I got to know by reading the above-mentioned web page.

- Where changing the query is needed then named queries can be used. That kind of queries can be defined using **`@NamedQuery(name = "qname", query="query")`** annotation. Example for a named Query is as follows.

```
@NamedQuery(name="findAllCustomersWithName",
query="SELECT c FROM Customer c WHERE c.name LIKE :custName")
```

Regarding above example later when we are creating a Query Object using the named Query `findAllCustomersWithName` parameter `custName` can be set.

I got a real life example of the way Named Queries are used from the above web page. This page made me clear the usage of Named Queries. I will be using these when developing my entity classes.

When further browsing through java persistence API I found a way to specify an array of Named Queries.

Java EE API (2007) *javax.persistence Annotation Type NamedQueries* [Online].
Available from:
http://java.sun.com/javaee/5/docs/api/javax/persistence/NamedQueries.html
[Accessed: 15 August 2009]

- If I have a quite possible number of named queries I can get the help of ***@NamedQueries*** annotation. According to the above page I created a possible usage example of Named Queries.

```
@NamedQueries( {
        @NamedQuery(name = "findByEmpId", query = "SELECT b FROM
BasicDAO b WHERE b.empId = :empId"),
        @NamedQuery(name = "findByFname", query = "SELECT b FROM
BasicDAO b WHERE b.fname = :fname")
})
```

This will help me when I am tackling with a lot of Named Query s in my entity classes. I think this will improve the readability of my codes.

I had some data fields in my tables having data type as “DateTime”. I wanted to find out how I will map those fields in to entity class variables.
My question was when mapping those columns if I use `java.util.Date` class as the data type will it work or not: so I searched the java API & came across the below mentioned two web pages.

Java EE API (2007) *javax.persistence Annotation Type Temporal* [Online]. Available from: http://api.dpml.net/javax/persistence/1.0/javax/persistence/Temporal.html
[Accessed: 16 August 2009]

Java EE API (2007) *javax.persistence Enum TemporalType* [Online]. Available from: http://www.jcourse.cn/docs/javaee5/javax/persistence/TemporalType.html
[Accessed: 16 August 2009]

The below I am mentioning what I got from the above sites.

- When the instance variables are of type `Date` or `Calendar` then **`@Temporal(TemporalType.type)`** annotation has to be specified. Here three values can be passed as type.

`TemporalType.DATE`	maps as	`java.sql.Date`
`TemporalType.TIME`	maps as	`java.sql.Time`
`TemporalType.TIMESTAMP`	maps as	`java.sql.TimeStamp`

So that I understood way to tackle with `Date` & `Calendar` type variables. Since I have Date type variables in my entity classes as well this information is helpful for me to tackle with Date type variables. I will be using **`@Temporal`** annotation as specified.

2.4 Automatic Generation Of Entity Classes

While searching on the web I found an interesting article about Java Persistence in NetBeans.

JavaBeat (2009) *JPA in NetBeans 6.1*[Online]. Available from: http://www.javabeat.net/articles/81-jpa-in-netbeans-61-1.html [Accessed: 21 October 2009]

The above web page contains a procedure to set up a connection to a MySQL database from NetBeans IDE.

Setting up a connection to a MySQL database from NetBeans IDE

1. Expand the service window in database node.
2. Right clicking *MYSQL* server node. So *MYSQL* server properties dialog box gets opened.
3. Fill the required properties & click ok.(The *MYSQL* database server should be running at this time.)
4. Right click MySQL server node & choose start.
5. Expand the server node
6. Right click on the database that needs to be connected & select connect. From the dialog box appearing enter the user name, password & click ok.

I followed above instructions & created a connection to My SQL database from Net Beans. It was successful.

Then I got the taste of ease & went on reading the article further. Then It was mentioned how to create entity classes by just three or four clicks.

JavaBeat (2009) *JPA in NetBeans 6.1*[Online]. Available from: http://www.javabeat.net/articles/81-jpa-in-netbeans-61-3.html [Accessed: 21 October 2009]

Generating a Entity class

1.Right click on the project & select “Entity classes from database”.
2. Select the database connection from the drop down menu.
3. Then available tables will get listed. Select the needed tables & add those to Selected tables.
4. Then specify the names of the entity classes. If Named Query annotations are needed tick the check box “Generate Named Query Annotations for Persistent Fields”.
5.Click the “Create Persistence Unit” button. Then create persistence unit wizard will open. There change Table Generation Strategy option to Create.
6.Click finish.

Here the article explains the auto generation of entity classes in a normal J2SE application but the procedure is still valid for an enterprise application.

I went through the above steps & had a real experience how easily entity classes can be auto generated. This was very efficient, saved my time, and increased productivity of my work. So at the time itself I generated required entity classes with the help of NetBeans. This procedure auto generated persistence.xml as well as Named Queries for all the fields in each class. No words to explain how much this article increased my productivity.

2.5 Deployment problems in Entity Classes

When I was going to deploy my entity classes the following error message was occurring.

```
"Operation 'pingConnectionPool' failed in 'resources' Config Mbean.
Target exception message: Class name is wrong or classpath is not set
for:com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource"
```

I was searching the root course for this error. Truly speaking I went through a lot of web sites, blog entries, and articles. I was really thinking some times hours how I could solve this problem. At last I got a valuable blog entry with a good explanation about the root course.

Reference:
Gerald. (2007).HOWTO: JSAS (Glassfish) with MySQL (updae). [Online] September 10th 2007. Available from: innoq.com
http://www.innoq.com/blog/gs/2007/10/howto_get_jsasglassfish_with_m.html
[Accessed: 12th December 2009].

- It will be impossible to make the entity classes work if the jdbc-driver is missing. If so following error message will occur in the case of MySQL.

```
"Operation 'pingConnectionPool' failed in 'resources' Config Mbean.
Target exception message: Class name is wrong or classpath is not
set for:com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource"
```

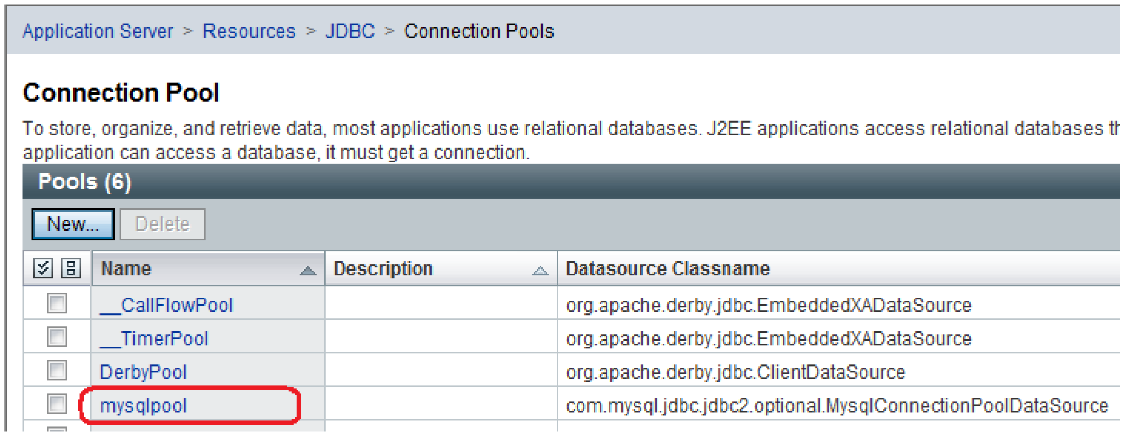
- The driver file for MySQL is `mysql-connector-java-5.1.5-bin.jar`. Driver file has to be moved to the directory `"%J2EE_HOME%\domains\domain1\lib\ext\"`.

- Then a connection pool has to be created using following settings.

```
Name: AnyName
Resource Type: javax.sql.DataSource
Database Vendor: mysql
Data Source Class Name:
com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource
serverName: 127.0.0.1(localhost)
port: 3306 (default port)
user: (user having permissions to connect mysql db)
password: (password)
databaseName: (the name of the database need to be connected)
```

- Add a jdbc resource. Then specify it in persistence.xml under the *`jta-data-source`* element.
  `Ex:    <jta-data-source>%JNDI_NAME%</jta-data-source>`

This is what was mentioned in the blog. I had that particular driver file with me but it was not in the directory the blog says. I copied the driver file to the directory it mentioned. After that everything was working well. I understood that Net Beans IDE has auto generated the connection pool & jdbc resource. I opened the Sun Java System Application Server Administration Console & checked it. I saw the created connection pool as well as JDBC resource.



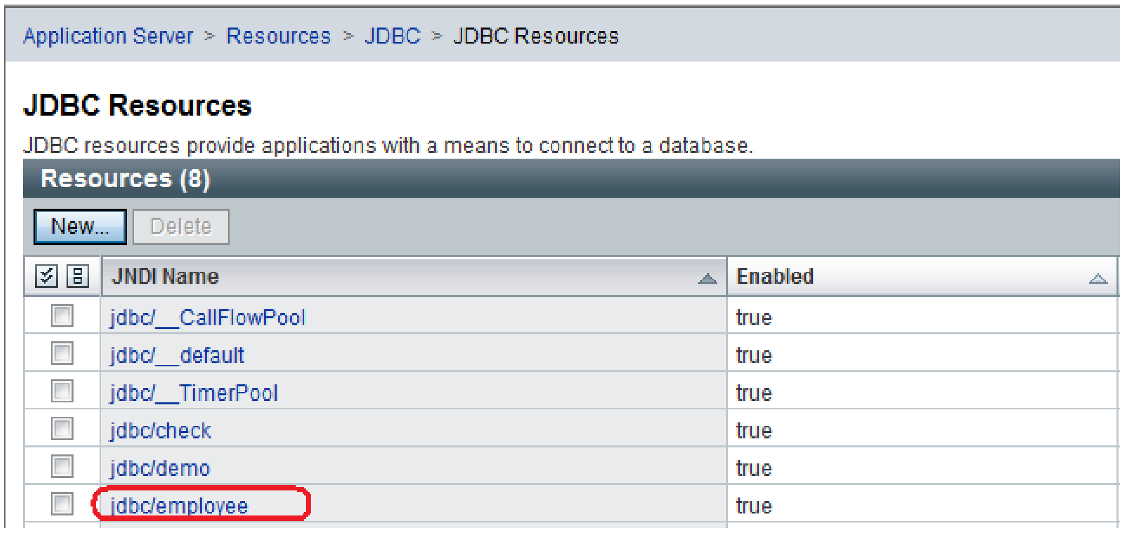
Connection Pool

To store, organize, and retrieve data, most applications use relational databases. J2EE applications access relational databases th
application can access a database, it must get a connection.

Pools (6)

	Name	Description	Datasource Classname
☐	__CallFlowPool		org.apache.derby.jdbc.EmbeddedXADataSource
☐	__TimerPool		org.apache.derby.jdbc.EmbeddedXADataSource
☐	DerbyPool		org.apache.derby.jdbc.ClientDataSource
☐	mysqlpool		com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource

Figure 5: Created Connection pool


Application Server > Resources > JDBC > JDBC Resources

JDBC Resources

JDBC resources provide applications with a means to connect to a database.

Resources (8)

	JNDI Name	Enabled
☐	jdbc/__CallFlowPool	true
☐	jdbc/__default	true
☐	jdbc/__TimerPool	true
☐	jdbc/check	true
☐	jdbc/demo	true
☐	jdbc/employee	true

Figure 6: Created JDBC Resource

At last I solved the error as well as I came to know underlying things going on. This article was helping me deploying & making my entities work since it helped me to solve error occurring when deploying.

2.6 Defining Session Beans

I wanted to have a better understanding about session beans. So I went through some parts of an OU course material.

Reference:
M362 *Developing concurrent distributed systems* (2008) Unit 7, The Open University. p. 45-53

I have mentioned below what I learnt regarding defining Session Beans from reading above-mentioned pages of the OU book.

- Session beans are enterprise components handling business functions. Clients(In my case servlets) call the declared business methods & get a response. Each client will use a distinct session bean & session beans are not persistent as well.

- Two types of session beans are found stateless session beans & stateful session beans.

  Stateless session beans: In this type client state (client specific detail) is not saved among multiple method calls. This type of beans is suitable for business methods not dependent on the invoking client.

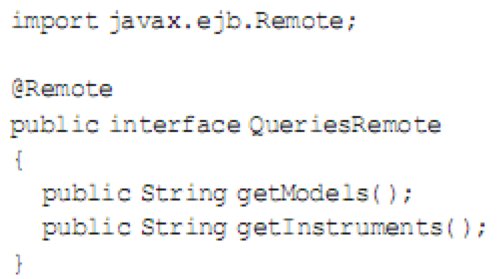
  Stateful session beans: In this type client state (client specific detail) is saved among multiple method calls. Here one stateful session bean can serve only to one client at a time.

- All the classes, interfaces, annotations related session beans are in `javax.ejb` package

- In both type of beans the client view is specified using a **business interface**. The methods accessible to the clients are declared there. Further more there it's mentioned whether clients can access the bean remotely (client & the bean are executing in different JVMS) or locally (both in same JVM).

- Business interfaces should be annotated as **`@Remote`** or **`@Local`** according to type of its access. Further more methods can't be *static* or *final*. Those should be *public* & should not start with *ejb*. Methods don't throw **`RemoteExcption.`**

- Small example of a remote Interface taken from the source



```
import javax.ejb.Remote;

@Remote
public interface QueriesRemote
{
  public String getModels();
  public String getInstruments();
}
```

Figure 7:Remote Interface example (taken from source)

- In the case of a stateless bean a class implementing the methods in the business interface has to be defined. The class should be annotated as ***`@Stateless`***.

- In the case of stateful beans also a class implementing the methods in the business interface has to be provided. The class should be annotated as ***`@Stateful`***. Further more one or more constructors also should be provided for the initiation of the session. There should be a method to signal the end of a session annotated as ***`@Remove`***.

My Client is an html browser based client & only web-based clients are allowed to interact with the system. I want to maintain small amount of client specific data only between servlets & JSPs. Further more I don't need to store client specific data after he/she logs out. Due to the above-mentioned facts I am not going to use Stateful session beans. Just an **`HttpSession`** object will be enough for my purposes. I am talking about that in the latter part. So reading about Stateful session beans did not help me for my work.

Anyhow I will be using Stateless session beans to handle business functions of my proposed solution. Defining business interfaces, Creating implementation Stateless session beans made very easy for me with the help of above facts.

2.7 EJB Security annotation

I was going through java EE API to find out how to apply security measures to session bean methods. By the way I found a page describing security related annotations.

Reference:
Java EE API (2009) *Package javax.annotation.security* [Online]. Available from: http://java.sun.com/javaee/6/docs/api/javax/annotation/security/package-summary.html
[Accessed: 10 December 2009]

By going through the above page I came to know the below mentioned information.

- All the security related annotations are found in **`javax.annotation.security`** package.

- Different type of security related annotations that can be found are mentioned below.

`@DeclareRoles`	used to declare security roles
`@DenyAll`	used to disallow all security roles from invoking methods
`@PermitAll`	used to allow all security roles to invoke particular methods
`@RolesAllowed`	used to mention list of roles allowed to invoke methods.
`@RunAs`	used to define the identity of an application

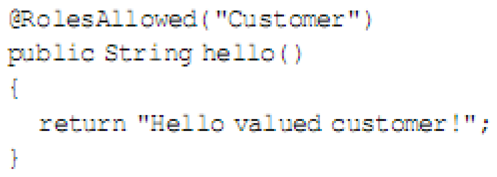
The above-mentioned annotations I would use for my methods since I wanted some methods to be accessible only for a certain group. This was helpful to understand the security related annotations but I needed some further more worked example. So I went through an activity & some pages on an OU book.

Reference:
M362 *Developing concurrent distributed systems* (2008) Unit 10, The Open University. p. 64-65

I read those two pages. Furthermore on the page 64 of this book there was an activity called "Activity 10.10 EJB annotation ". I went through the instructions that were given for that activity as well.

The following facts were very clear for me after doing so.

- In my sources two examples for @RolesAllowed annotation are found one used for a class & other used for a method. I am just taken the below extract from the example for a method.



```
@RolesAllowed("Customer")
public String hello()
{
  return "Hello valued customer!";
}
```

- After specifying allowed roles using ***@RolledAllowed*** annotation it has to be stated which of users are in the particular role. To do that ***security-role-mapping*** element has to be added to the ***sun-ejb-jar.xml***. If Net Beans is used it can be found under configuration Files sub section of the project. Below I am showing an example of a sun-ejb-jar.xml file taken from the source. Newly added elements are highlighted in blue color.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD
Application Server 9.0 EJB 3.0//EN"
"http://www.sun.com/software/appserver/dtds/sun-ejb-jar_3_0-
0.dtd">
<sun-ejb-jar>
    <security-role-mapping>
        <role-name>Customer</role-name>
        <group-name>customers</group-name>
    </security-role-mapping>
  <enterprise-beans/>
</sun-ejb-jar>
```

These two sources were very helpful for me when applying security measures to session beans methods. While the first resource giving me theoretical information second resource gave me a lot of practical details. So with the help of these two resources I made some methods accessible for only a group of people. So security measure I wanted to apply was achived.

2.8 Entity management from the session bean

I got to know about the way entity management is achived from the session beans with the help of an OU book.

Reference:

M362 *Developing concurrent distributed systems* (2008) Unit 7, The Open University. p. 37-53
I came to know about the following facts by reading those pages.

- **Persistence Context** is a collection of entities, which the run time system knows their persistent storage.
- For entity management such as creating an entity & making it persistent, storing the changes done to entities using getter & setter methods in the database, deleting data from the database a **persistence context** is needed.
- Putting entities into persistence context, marking entities for removing from the persistence context can be done using an **Entity Manager** (actually using methods provided in the **javax.persistence.EntityManager**).

- To create an entity manager ***`persistence unit`*** has to be defined in `persistence.xml` file.
- Here in session beans we can delegate the job of creating an entity manger to the ejb container.(a container managed entity manager).
- Just annotating declaration of required object will do the dependency injection. So that container creates an object(Entity manager) using the details at the annotation. As an example following code snippet shows how it's done.

```
@PersistenceContext(unitName="persistence_unit_name")
private EntityManager  manager;
```

- After that EntityManager can be used for managing entities.

Store a new entity in the database:

1. Create a new entity by calling its constructor. As an example I take a Book entity.
   ```
   Book object=new Book("bm4428","T.S.Eliot");
   ```
2. Call the persist method of the EntityManager.
   ```
   manager.persist(object);
   ```

Finding entities:

- Finding by primary key

  Entities can be found by primary key using find method in EntityManager interface. Entity class & primary key value has to be passed as parameters to that method. Since classes can't be passed as parameters to methods a special object representing class is passed (static variable named class). As an example if entity class is Book & primary key is "A0002" then usage is:

  ```
  Book object=manager.find(Book.class,"A0002");
  ```

- Using Queries

  `javax.persistence.Query` type object can also be created using **`createQuery (String query)`** method in EntityManager interface as well. Here query has to be a Java Persistence query language statement. After that the help of **`getResultList()`** Method can be taken as in the previous scenario.

  So as a simple example:

  ```
  Query query=manager.createQuery("SELECT b from Book b");
  List<Book> objects=(List<Book>)query.getResultList();
  ```

Deleting the entities

After finding entities as above just use the help of `remove(Objet entity)` method in the `EntityManager` interface. As an example if the object that has to be removed is called garbage then `manager.remove(garbage)` will work

The above information was really helpful for me when I am coding the part of managing entities from the Session beans since I want to manage entities in the mid of the business functions in Session Beans. How to store a new entity in to a database, how to search an entity from the database, deleting an entity from the database I got to know from the above book. The coding, underlying architecture of managing entities was very clear. I have coded 90% of the entity management coding with the help of this.

Further more I wanted some real life examples of how named Queries are used for entity management. So I had a look on Java API.

Reference:
Java EE API (2007) *javax.persistence Interface EntityManager* [Online]. Available from: http://java.sun.com/javaee/5/docs/api/javax/persistence/EntityManager.html

[Accessed: 13 December 2009]

Java EE API (2007) *javax.persistence Interface Query* [Online]. Available from: http://java.sun.com/javaee/5/docs/api/javax/persistence/Query.html

[Accessed: 13 December 2009]

Using named queries:

`javax.persistence.Query` type object can be created using `createNamedQuery (String queryName)` method in EntityManager interface. Then with the help of `setParameter(String name, Object value)` method declared in `Query` interface arguments can be bound to named parameters. After that with the help of `getResultList()` method in the *`Query`* interface itself the named query can be executed & result can be stored in a `List<Object>`.

Let's assume that there is a named query hotQuery in Book entity class & query has a parameter called id. Then the coding snippet becomes like below. I created some prototype cording after going through the above web pages. It worked.

```
Query query=manager.createNamedQuery ("Book.hotQuery");
query.setParameter("id",”A0002”);
List<Book> objects=(List<Book>)query.getResultList();
```

These two web sites were very much helpful for getting the maximum use of defined named Queries. I used the pre-defined methods I have mentioned above in the business methods in the session beans. It was very efficient using the Named Queries in the above way.

-

2.9 Declaring Message Driven Beans

When implementing the functionality to allow the employees to inform if they find that some of their data should be changed, I wanted to know about message driven beans, how I can implement it. So I got that information from a NIIT course book. It is talking about the Message Driven Beans 2.1 but after reading it concept was clear to me.

Reference:
Quarter 8 *Developing Enterprise-wide Applications using J2EE Technologies Part 3 -II* (2008) Lesson 4 B, NIIT. p. 4B.3 - 4B.28

After reading the above lessons the below mentioned facts were very clear to me.

Asynchronous messaging between two java bean components is achieved using message driven beans. On receipt of a message they can perform different actions such as calling another bean's methods, perform some business operations.

Message driven beans internally use **`javax.jms API (Java Messaging Services API)`** for messaging. In `JMS` the message sender is called "`JMS Producer`". Message Receiver is called "`JMS Consumer`". There's a `middle layer` between the `JMSProducer` & `JMSConsumer`. `JMS producer`'s job is to send the message to the `middle layer`. `Middle layer` take the responsibility of sending the message to the `JMS Consumer`. So that JMSProducer doesn't have to wait for message confirmation & performance will get increased.

Two messaging models are supported by JMS.

1. Publisher Subscriber model.

   In this model multiple message senders send message to the middle layer. In this model middle layer is called a Topic. Multiple message consumers can subscribe to the Topic. So that message will be delivered to all subscribed message consumers.

2. Point To Point Communication:

   In this model middle layer is called a Queue. There massage transferring occurs between only within one message producer & one message consumer.

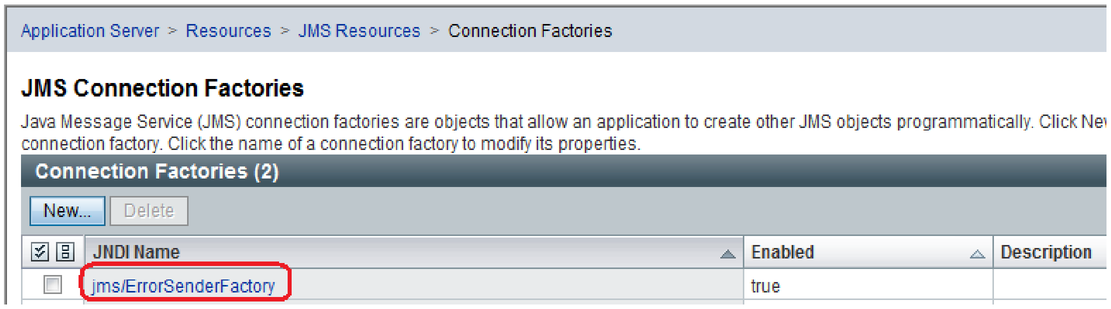
There should be a class that implements **`javax.jms.MessageListener`** interface. In that interface only one method is declared.

`Public void onMessage (Message message)`. So that in that class this method should be implemented. `javax.jms.Message` interface has a sub interface called TextMessage. That also can be used when handling messages that contain text.

For the bean to work two administered objects, ConnectionFactory & JMS destination Recourse has to be created using J2EE admin console but Net Beans does that on behalf of the user.

So as in publisher subscriber model I don't want to have lot of message producers & message consumers in my scenario. In my solution I need only one message producer & one consumer. After reading above I came to a decision that for my solution I am using the Point-to-Point Communication model since in my case I just want the errors to be entered to the database when the employee reports errors. I can implement it in such a way that when the message arrives to the message driven bean it stores the error to the database.

Above-mentioned chapter of the book helped me to take that decision clearly. I created administered objects as well with the help of the above book. I am giving below two screen shots of created Connection factory type object as well as Destination type object.


Application Server > Resources > JMS Resources > Connection Factories
New...
Delete

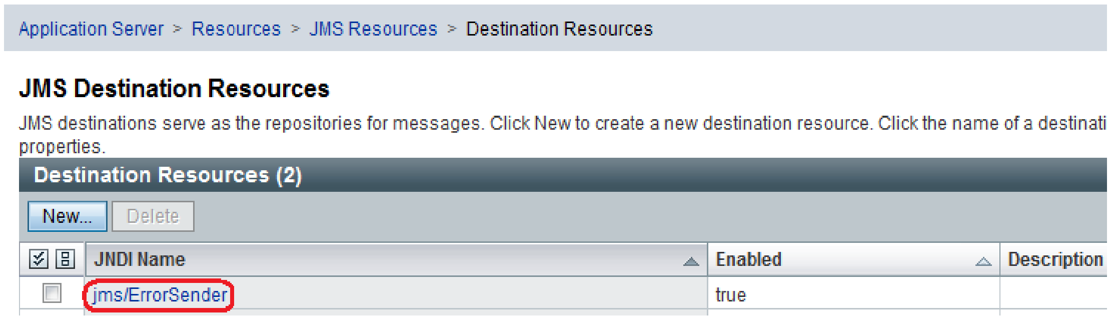
JMS Connection Factories

Java Message Service (JMS) connection factories are objects that allow an application to create other JMS objects programmatically. Click Ne
connection factory. Click the name of a connection factory to modify its properties.

Connection Factories (2)

JNDI Name	Enabled	Description
jms/ErrorSenderFactory	true	

Figure 7: Connection Factory type Object


Application Server > Resources > JMS Resources > Destination Resources
New...
Delete

JMS Destination Resources

JMS destinations serve as the repositories for messages. Click New to create a new destination resource. Click the name of a destinati
properties.

Destination Resources (2)

JNDI Name	Enabled	Description
jms/ErrorSender	true	

Figure 8: Destination type Object

Further more I got an understanding about JMS API, how to code my Message driven bean as well. Since it was about Message Driven Beans 2.1 before start developing message driven bean I wanted to find out annotations related Message driven beans in new EJB 3.0.

So I came across an article related annotations used in EJB 3.0.

Reference:
ORACLE (2008) *EJB 3.0 Metadata Annotations Reference* [Online]. Available from: http://download.oracle.com/docs/cd/E13222_01/wls/docs100/ejb30/annotations.html#wp1416916
[Accessed: 15 December 2009]

The annotations that have to be used when designing a Message Driven Bean was very clear to me after reading the above extract. I have mentioned below what I got from the above web page.

The Message Driven bean class should be annotated using **`@MessageDriven`** annotation. According to what the page says I created a small example for the annotation.

```
@MessageDriven(mappedName = "JNDINameofDestinationResource ",
activationConfig =  {
  @ActivationConfigProperty(propertyName    =    "destinationType",
  propertyValue = "javax.jms.Queue")
 })
```

`mappedName` attribute of the **`@MessageDriven`** annotation is here to specify a product-specific name to which message-driven bean is mapped. JNDI name of the message driven bean can be mentioned here.

`@ActivationConfigPropery` annotation is used for the configuration of the message driven bean in operational environment. Some properties it may include are acknowledgement modes,message selectors,endpoint types etc.

After going through these Literature Items I started developing my Message Driven Bean. Due to the information I gathered above I was capable of developing Message Driven Bean for error notification.

2.10 Developing Servlet classes

I was collecting servlet & JSP related concepts from the following book.

Basham, B., Sierra, K. & Bates, B., (2008). *Head First Servlets and JSP*. Sebastopol: O'Reilly Media Inc.

I am just mentioning the things I paid my attention while I was going through the book & at the mean time I will be discussing the help I got from those concepts.

- Web browsers make http requests from the browser to the server mainly using two ways GET & POST. If GET is used when sending parameters to server, parameters are appended to the URL as a query string. If these parameters contain confidential data such as usernames, passwords then those data get visible. It's a security risk.

  So that good solution for that problem is using POST request where parameters are added to the http request body & sent. Then data won't become visible. I decided that in my proposed solution also I would be handling with post requests to have a certain amount of security.

- Way To Create a Servlet Class to handle http post requests.

  1. A class extending **`javax.servlet.http.HttpServlet`** class should be created.
  2. Then **`public void doPost (HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException`** method should be implemented. The required actions to perform should be mentioned in sided the method body.

  Since it explains the simplest possible way to develop a Servlet class I also developed my servlet exactly according to above procedure.

- HTTP is a stateless protocol. In the sense each request is independent from other. Sometimes some data need to be available within multiple http requests. In other words sometimes maintaining a session is needed.

  So in my case also I want to store a small amount of client specific data within Servlets & JSPs. Further more I just want to store some client specific data while users browse the pages only. Long time storage of client specific data is not required. So I am simply using an HttpSession object to maintain client specific data

-

Way to maintain a session

1. After executing `getSession()` method of an `HttpServletRequest` object a `javax.servlet.http.HttpSession` type object can be returned.
2. Then `setAttribute(String name,Object value)` method of the `HttpSession` type object can be used to bind an object to the session with a particular name.
3. `getAttribute(String name)` method of the `HttpSession` type object can be used to return the object bounded with that name previously.

   The below simple example further illustrates the concept: (Bounding a String object with a name "name", Retriving the "name" attribute later)

   ```
   HttpSession session=request.getSession();
   session.setAttribute("name","Melroni");
   String name=(String)session.getAttribute("name");
   ```

I have come to know how to maintain an HttpSession, how to store data in a Session, How to retrieve data values stored in the Session object. I am using this procedure when saving Client state within my Servlets & JSP s as well.

I got some information about the servlets from the below mentioned OU course book as well.

Reference:

M362 *Developing concurrent distributed systems* (2008) Unit 8, The Open University. p. 43-52

Forwarding a Http Request to a Java Server Page

This can be achieved with the help of `javax.servlet.RequestDispatcher` interface. RequestDispatcher type object can be obtained from the `HttpServletRequest` type object with the help of `getRequestDispatcher(String path)` method. This is how it's done.

If we assume that we want to forward the request to a JSP page called `ErrorConfirm.jsp`:

```
RequestDispatcher
disp=request.getRequestDispatcher("/employee/ErrorConfirm.jsp");
```

Then we can call the `forward (HttpRequest req, Httpresponse res)` method to forward the request to a Java Server Page.

```
disp.forward(request, response);
```

After calling forward method it should not be tried to add anything to response stream since forward method delegates response completely to the destination.

There's another method called `include(HttpRequest req,HttpResponse res)`. If it's used part of the response generation is forwarded to the destination. First servlet can still add items to response even after invoking the method.

I wanted to forward the presentation part from servlets to Java Server pages. So I got the maximum use of this information to forward the output from a Servlet to a Java Server page for presentation. Further more most of the time I used **`forward()`** method since I was forwarding the output to the display at the end after finished all the controlling parts.

Accessing Session Bean methods from the Servlet

Accessing a Session Bean from the servlet can be achieved by dependency injection with the help of **`@EJB`** annotation in **`javax.ejb`** package. For the servlet only the session bean's business interface is visible. As an example if business interface is **`CartLocal`** then:

```
@EJB
private CartLocal shoppingCart;
```
will do the job.

Then methods declared in the session bean's business interface could be accessed.

I also wanted to access the methods declared in the session beans from servlet classes. I used just @EJB annotation & did that. After I retrieved the interface type object it was normal.

-

2.11 Sending message To a Destination(Topic/Queue)

I searched the information how to send a message to a topic. I got the required details from the below mentioned book. Earlier also I mentioned this chapter of a book but this is the second time I am reading it.

Reference:
Quarter 8 *Developing Enterprise-wide Applications using J2EE Technologies Part 3 -II* (2008) Lesson 4 B, NIIT. p. 4B.13 - 4B.18

I am mentioning below what I got to know about sending messages to a Destination.

- Here first client has to do a JNDI look up & find out **`javax.jms.ConnectionFactory`** object & **`javax.jms.Destination`** type object associated with the message driven bean. **`Desination`** type object can be of type **`javax.jms.Queue`** or **`javax.jms.Topic`**. When performing a lookup **`NamingException`** might occur. It has to be handled.

- If the JNDI name of the ConnectionFactory is "**`jms/SenderFactory`**" &

  Destination is a Queue & it's JNDI name is "**`jms/ErrorSender`**" then coding should be like below.

  ```
  InitialContext jndiContext=new InitialContext();
  ConnectionFactory
  Fact=(ConnectionFactory)jndiContext.lookup("jms/SenderFactory");
  Desination dest=(Queue)jndiContext.lookup("jms/ErrorSender");
  ```

- All the following methods throw `JMSException`.

  Then a **`javax.jms.Connection`** type object can be obtained by invoking **`createConnection()`** method on the returned **`ConnecionFactory`** type object.

  ```
  Connection con= Fact.createConnection();
  ```

- Then a **`javax.jms.Session`** type object can be achived using **`createSession(boolean transacted, int acknowledgeMode)`** method of the javax.jms.**`Connection`** interface.

  Example:

  ```
  Session session=con.createSession(false,Session.AUTO_ACKNO
  WLEDGE);
  ```

- Then a **`javax.jms.MessageProducer`** type object can be achived using **`createProducer(Destination destination)`** method declared in the **`javax.jms.Session`** interface.

  Example:

  ```
  MessageProducer msgPrd=session.createProducer(dest);
  ```

- **`Session`** interface has a method called **`createTextMessage()`** **that returns a** **`javax.jms.TextMessage`** type object.

  Example:

  ```
  TextMessage msg=session.createTextMessage();
  ```

- **`TextMessage`** interface has a declared method **`setText(String string)`** to set the Sting part of the message.

- At last `MessageProducer` interface has a method **`send(Message message)`** used to send messages.

  ```
  msgPrd.send(msg);
  ```

In my proposed solution also I had to send messages to a Queue from my Servlet class. So I went through the sequence of steps in the procedure. One by one I also followed the above steps. I was able to code that part due to the above facts I found. At last I checked my proposed solution was functioning well since it was capable of sending the employee error messages to the Destination.

-

2.12 JDBC realm

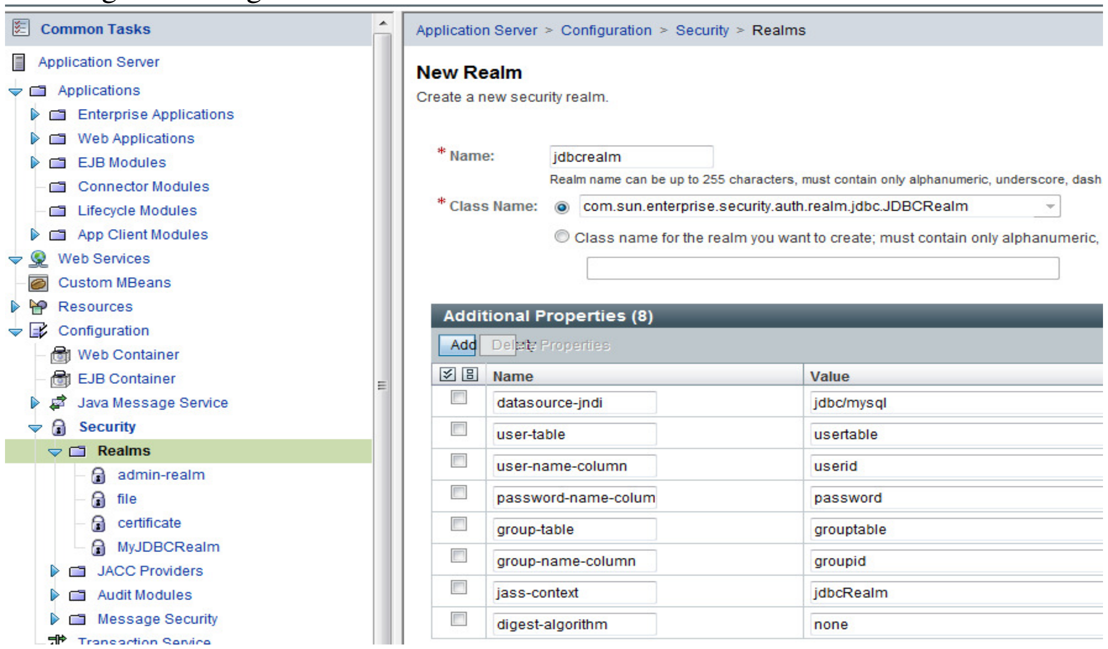
I wanted to use a combination of declarative & programmatic authorization in Servlets. To do that first of all I wanted to have a realm. So I selected to use a JDBC realm since I can directly get the user names, group names, and passwords from the database if JDBC realm is used. After making a realm I could specify it in my `web.xml` & `sun-web.xml` files to authorize the users. I found the details on how to create a JDBC realm from a blog spot but in the blog spot the writer has used a different version of the application server I was using anyhow the steps were still the same. So with the help of below information I managed to create a JDBC realm for my proposed solution.

Reference:
Chan, S. W. (2007).*JDBC Realm in GlassFish with MySQL*. [Online] April 23rd 2007. Available from: blogs.sun.com
http://blogs.sun.com/swchan/entry/jdbcrealm_in_glassfish_with_mysql
[Accessed: 12th December 2009].

In this article I pay my attention from the number 14 since I have already performed the previous options when I read this.

I had the database running, JDBC drivers working, group tables & user tables created accordingly. Further more Net Beans created connection pool & data sources on behalf of me at an earlier stage.

So I http://localhost:4848/asadmin/ & logged to server & created a JDBC Realm according to the blog.



New Realm
Create a new security realm.

* Name: jdbcrealm
Realm name can be up to 255 characters, must contain only alphanumeric, underscore, dash

* Class Name: (●) com.sun.enterprise.security.auth.realm.jdbc.JDBCRealm
( ) Class name for the realm you want to create; must contain only alphanumeric,

Additional Properties (8)

Name	Value
datasource-jndi	jdbc/mysql
user-table	usertable
user-name-column	userid
password-name-colum	password
group-table	grouptable
group-name-column	groupid
jass-context	jdbcRealm
digest-algorithm	none

Figure 9: Creating a JDBC realm

The blog writer has forgotten to mention the class Name that has to be used in the JDBC realm. Anyhow I found it in the second comment posted by a person called viperomega.

2.13 Mapping Security Constraints in web.xml/sun-web.xml

I wanted to come to a conclusion that how I should perform the web resource authorization. While I was thinking about that I remembered the web resource authorization activity on a OU course book.

Reference:

M362 *Developing concurrent distributed systems* (2008) Unit 10. Activity 10.9 web resource authorisation, The Open University. p. 62

According to the activity there are four steps to be taken

1. Setting the Login Configuration
2. Adding roles
3. Adding Constraints
4. Mapping groups & principles to roles

1.Setting the Login configuration

I followed the steps accordingly. I opened the web.xml file. I opened the Security tab. Then I specified the name of the JDBC realm I created in the Realm Name input field. The activity tells us to use “file” as the realm name to use file realm.

The activity explains about the Basic Authentication as well but it’s not suitable for me. Then I selected Form based authentication as the authentication mechanism since I wanted to clear the session when one person logs out. For that I specified two Java server pages for the Form Login page as well as the form Error page.

2.Adding security roles

As it is mentioned in the activity I expanded the security roles section & I also added two security roles in my proposed solution `employees` & `managers`.

3. Adding Constraints

According to the activity I also browsed through the Security Constrains section & added three security constraints with following constraints

1. Resource name: `EmployeePages`

   URL Pattern: `/employee/*`

-

2. Resource name: `ManagerPages`

   URL Pattern: `/manager/*`

3. Resource name: `MainConstraint`

   URL Pattern: `/LoginController`

Then I applied the `EmployeePages` constraint to the role `employees.` Then only people with `employees` role name can access pages declared as `/employee/*`.

After that I applied the `ManagerPages` constraint to the role `managers.` Then only people with `managers` role name can access pages declared as `/manager/*`.

Finally I applied the `MainConstraint` constraint to the both roles `managers & employees. LoginController` is the first starting page of my solution. So that it's ensuring only one of above two role names can move further.

4. Mapping groups & principals to roles

In this case also I followed the procedure that was mentioned on the above activity. I opened sun-web.xml. For my two roles I applied the two-group names accordingly. I was not mapping principles to roles since my all users were in one of the group names `employee` group or `manager` group.

So that 's how I applied authorization to my proposed solution with the help of that activity. So that with the above mechanism here, annotation based security at session beans I think I was able to give a certain amount of security level to my proposed solution.

2.14 Java Server Pages

I wanted to know about generation of Java Server Pages. I didn't want to go for advanced features. In my solution Java Server Pages are just used for presentation of some output generated by another component most of the time a servlet. So I got some information about different tags used, some predefined objects used in JSP from the book previously I was referring for the Servlets.

Basham, B., Sierra, K. & Bates, B., (2008). *Head First Servlets and JSP*. Sebastopol: O'Reilly Media Inc.

JSP directive tags
Usage: when specifying global information about a Java Server Page.
Three directives can be found respectively page, include & taglib.
Examples:
1. `<%@ include file="temp.jsp" %>`
When including other pages within the present JSP page, when entering headers, footers this tag was effective as well as helping to me.

2.`<%@ page import="java.util. *" %>`
When importing java packages to the current JSP page this was helpful.

JSP scripting tags
1. Declaration tag `<%! int count = 0; %>`
Usage: when declaring & initializing instance variables I used this tag.
2.Scriplet tag `<% int count = 0; %>`
When I wanted some java coding to be executed at the run time of the JSP page I used this tag. This was the most frequently used tag I used for my solution.

3 JSP Expression tag `<%= count %>`
This tag contains a java expression that produces a value that is displayed as output.
When I wanted to display some value of a variable I always used this tag.

Further more I found some implicit objects that are used in Java Server Pages. Three implicit objects `request` that is of type `HttpServletRequest`, `response` of type `HttpServletResponse`, `session` type of `HttpSession` was really helpful. I used those in my JSP pages frequently. It was ease for me since without initializing I can just access the object.
With the help of all above-mentioned facts I developed the required Java Server Pages for my proposed solution.

3.Account Of Outcome

3.1Analysis

3.1.1 Naturalistic Observation

To identify the problem, it's context, the reasons for calling it a problem, what kind of a solution I can give to the problem I needed to gather data. I used three data gathering techniques for this purpose.

Naturalistic Observation: This technique I used to find out the context of the problem, how the current process is occurring. I went to all five places & spent a day at each place. I was observing how they perform the actions, but in this technique I couldn't find out the reason for observations. I was noting down every thing.

Using the details I gathered from the above technique I drew activity diagrams for the processes I observed. I am mentioning those diagrams below one by one.

- I observed head of another department came to the human resource department & asked a report about employees satisfying a certain criteria at the same place. I observed this in two places.

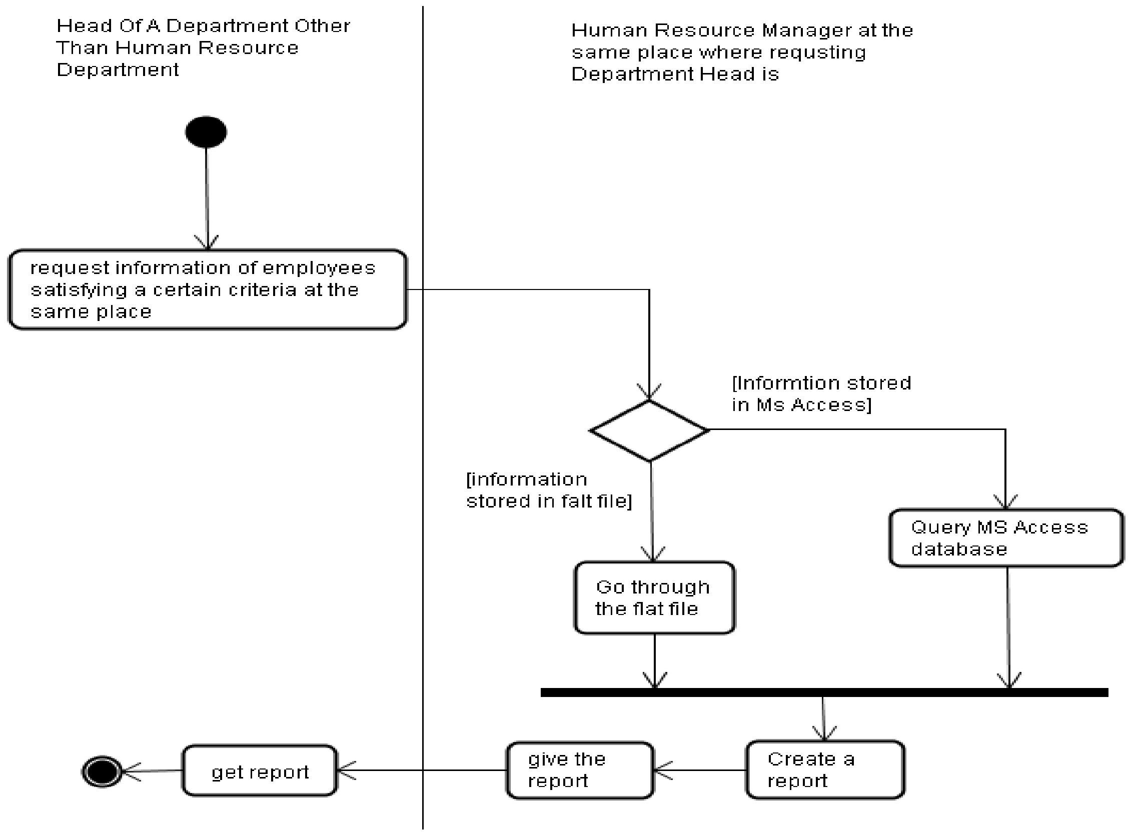

Head Of A Department Other Than Human Resource Department
Human Resource Manager at the same place where requsting Department Head is
request information of employees satisfying a certain criteria at the same place
[Informtion stored in Ms Access]
[information stored in falt file]
Query MS Access database
Go through the flat file
Create a report
give the report
get report

Figure 10: Activity Diagram 1(Existing System)

- In two places I observed that a head of the department come to the human resource department & asked for a report of employees satisfying certain criteria at another place. Then the human resource manager at the same place of the head gave a call to the employee resource manager where employee information was available. Then the human resource manager at that place created the report & faxed the report.

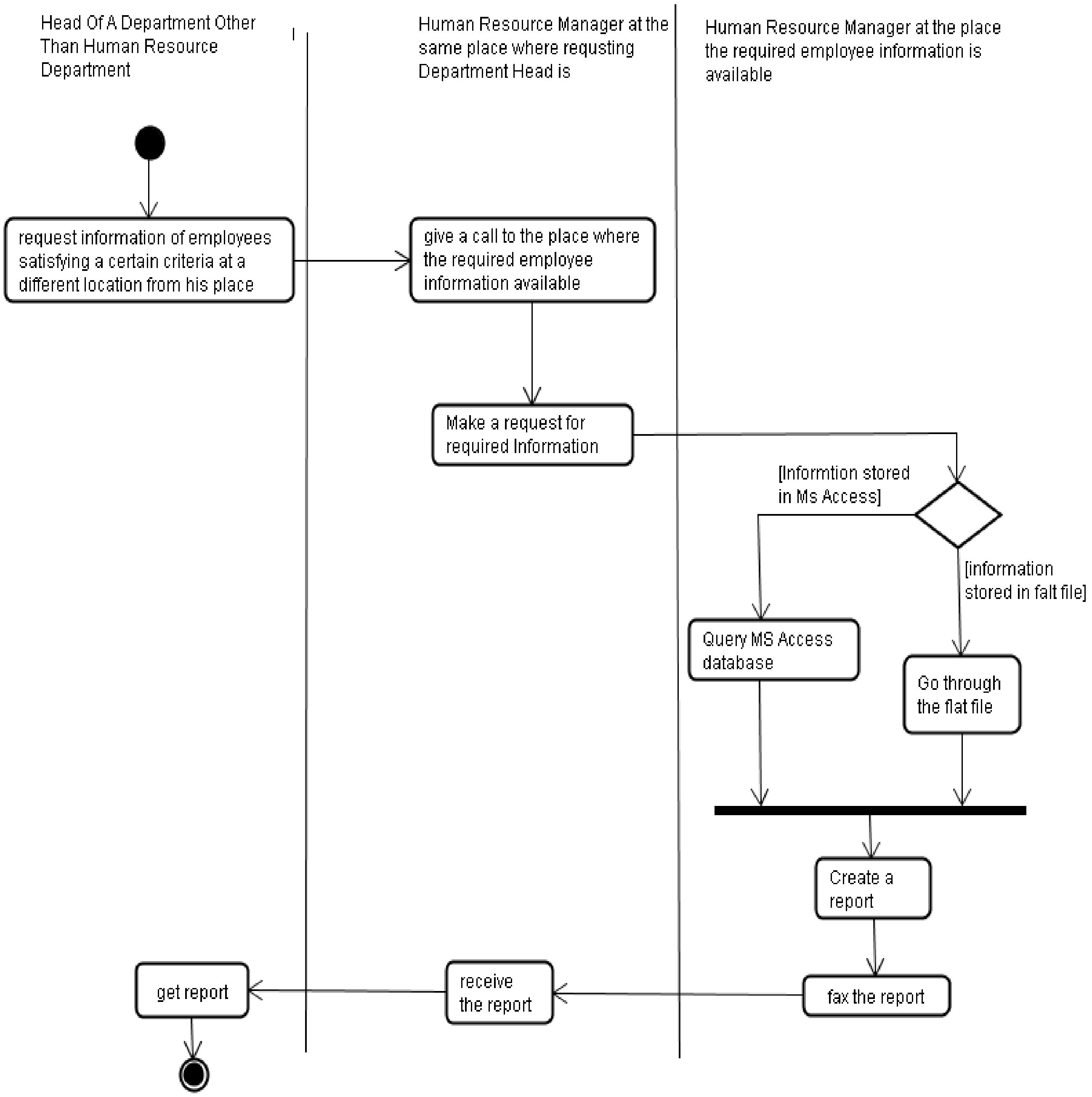

Head Of A Department Other Than Human Resource Department
Human Resource Manager at the same place where requsting Department Head is
Human Resource Manager at the place the required employee information is available
request information of employees satisfying a certain criteria at a different location from his place
give a call to the place where the required employee information available
Make a request for required Information
[Informtion stored in Ms Access]
[information stored in falt file]
Query MS Access database
Go through the flat file
Create a report
fax the report
receive the report
get report

Figure 11: Activity Diagram 2(Existing System)

-

- In one place I observed the head of a department came & asked the human resource manager to give a report of all the employees satisfying a certain criteria at all five places. Then the human resource manager called all other employee managers and got the reports faxed. Then he created the aggregated report by looking at all reports.

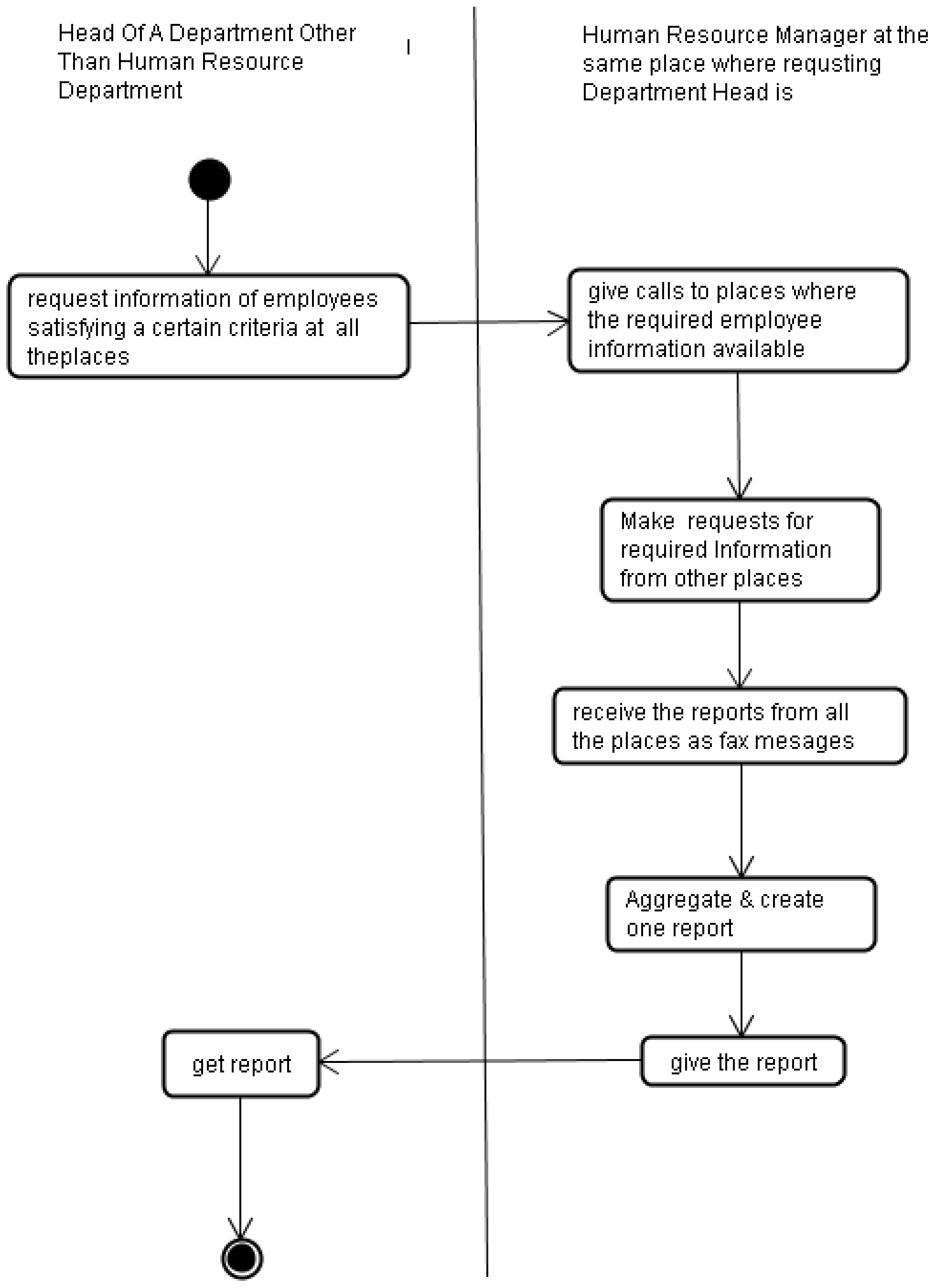

Head Of A Department Other Than Human Resource Department
Human Resource Manager at the same place where requsting Department Head is
request information of employees satisfying a certain criteria at all theplaces
give calls to places where the required employee information available
Make requests for required Information from other places
receive the reports from all the places as fax mesages
Aggregate & create one report
give the report
get report

Figure 12: Activity Diagram 3(Existing System)

- In three places I observed an employee wanting to know his information came, filled a form & handed it over to the human resource manager. Then manager created a report & handed over the report to employee. Here in the form the employee had to mention his employee ID number & had to sign as well.

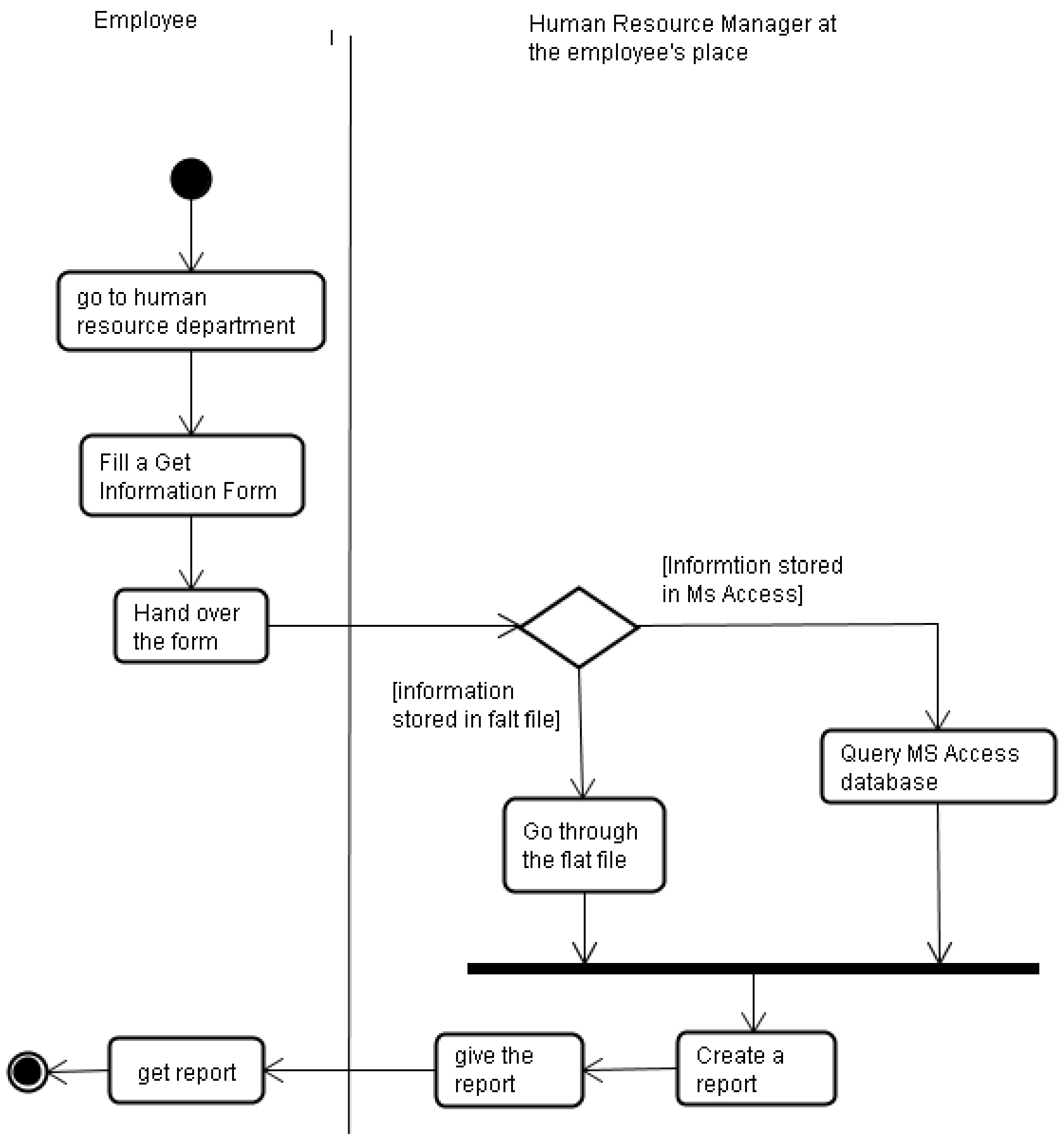

Employee
Human Resource Manager at the employee's place
go to human resource department
Fill a Get Information Form
Hand over the form
[Informtion stored in Ms Access]
[information stored in falt file]
Query MS Access database
Go through the flat file
Create a report
give the report
get report

Figure 13 Activity Diagram 4(Existing System)

- In two places I saw that one employee wanted to report the human resource manager about an error in his records. He came & informed the employee resource manager. Then human resource manager asked for his employee ID. After having a look at ID & the person twice, human resource manager decided that he is the correct person & did the changes. I understood that if the manager couldn't validate the ID manager wouldn't do the changes in the records.

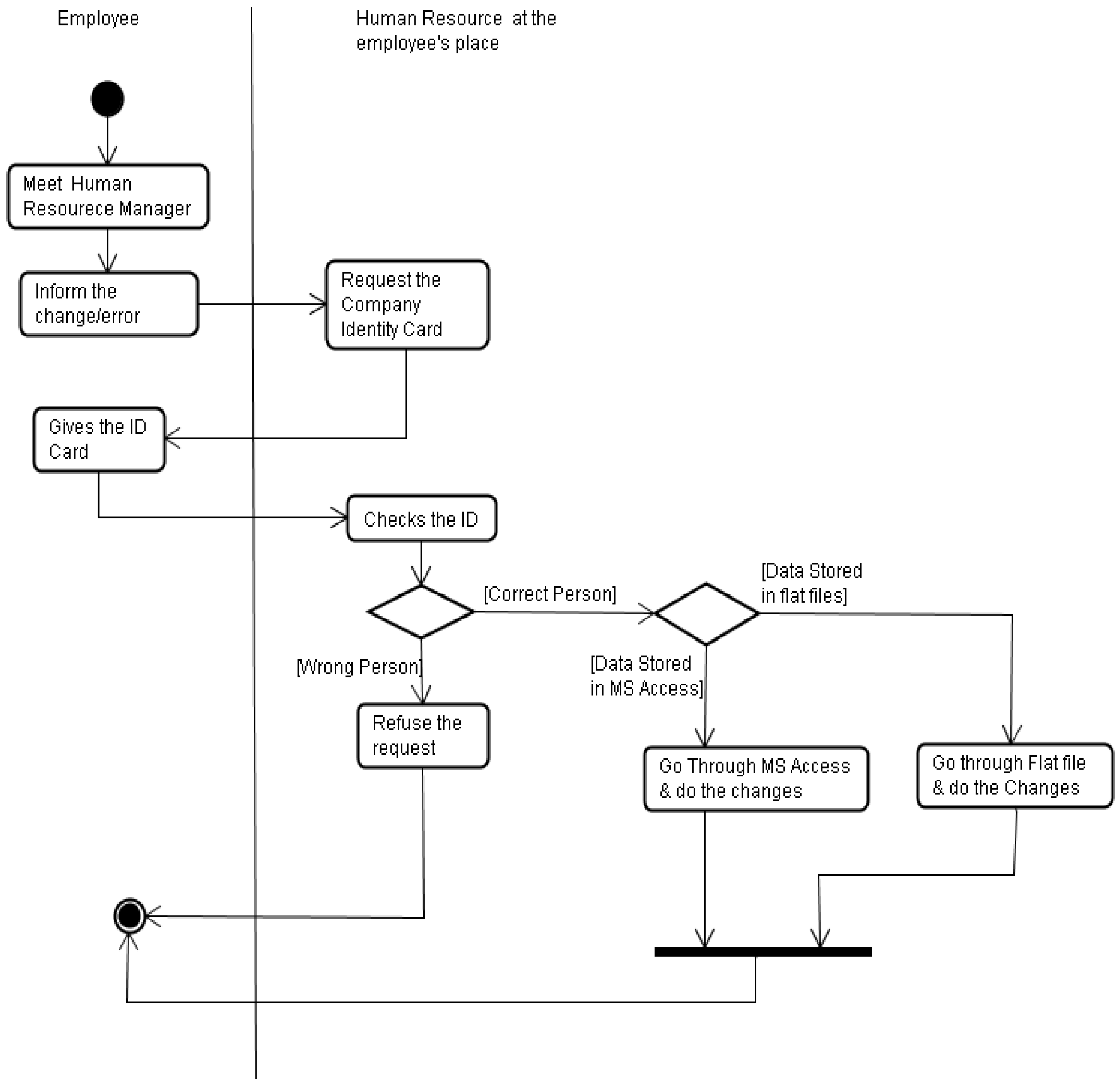

Employee
Human Resource at the employee's place
Meet Human Resourece Manager
Inform the change/error
Request the Company Identity Card
Gives the ID Card
Checks the ID
[Correct Person]
[Wrong Person]
Refuse the request
[Data Stored in flat files]
[Data Stored in MS Access]
Go Through MS Access & do the changes
Go through Flat file & do the Changes

Figure 14: Activity Diagram 5(Existing System)

- In three places I observed a new employee joining to the company. Human resource manager asked the employee information & entered data to the computer.

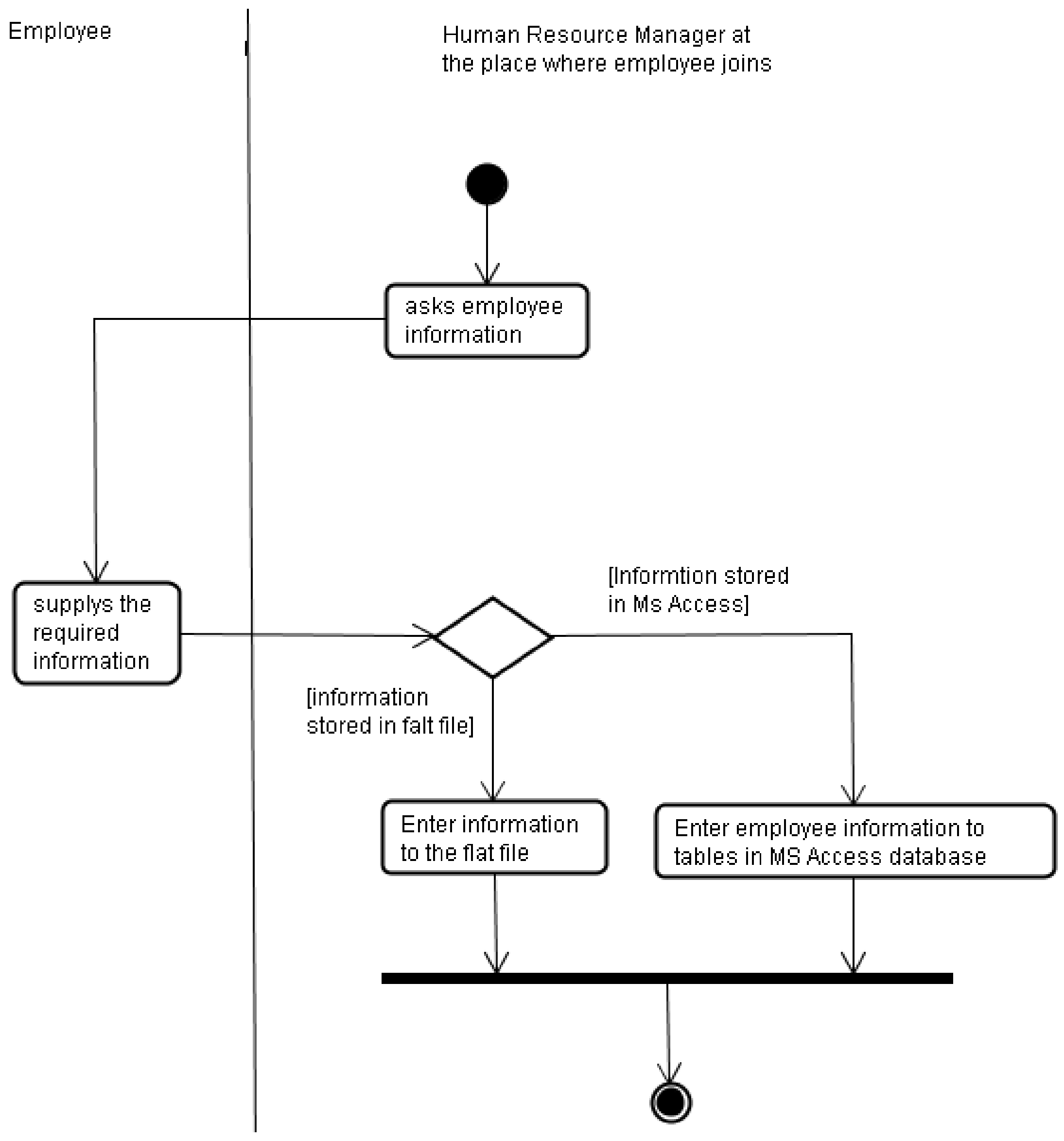

Employee
Human Resource Manager at the place where employee joins
asks employee information
supplys the required information
[Informtion stored in Ms Access]
[information stored in falt file]
Enter information to the flat file
Enter employee information to tables in MS Access database

Figure 15: Activity Diagram 6(Existing System)

- In two places I observed employees leaving the company. Human Resource Managers don't keep the data of the persons left.

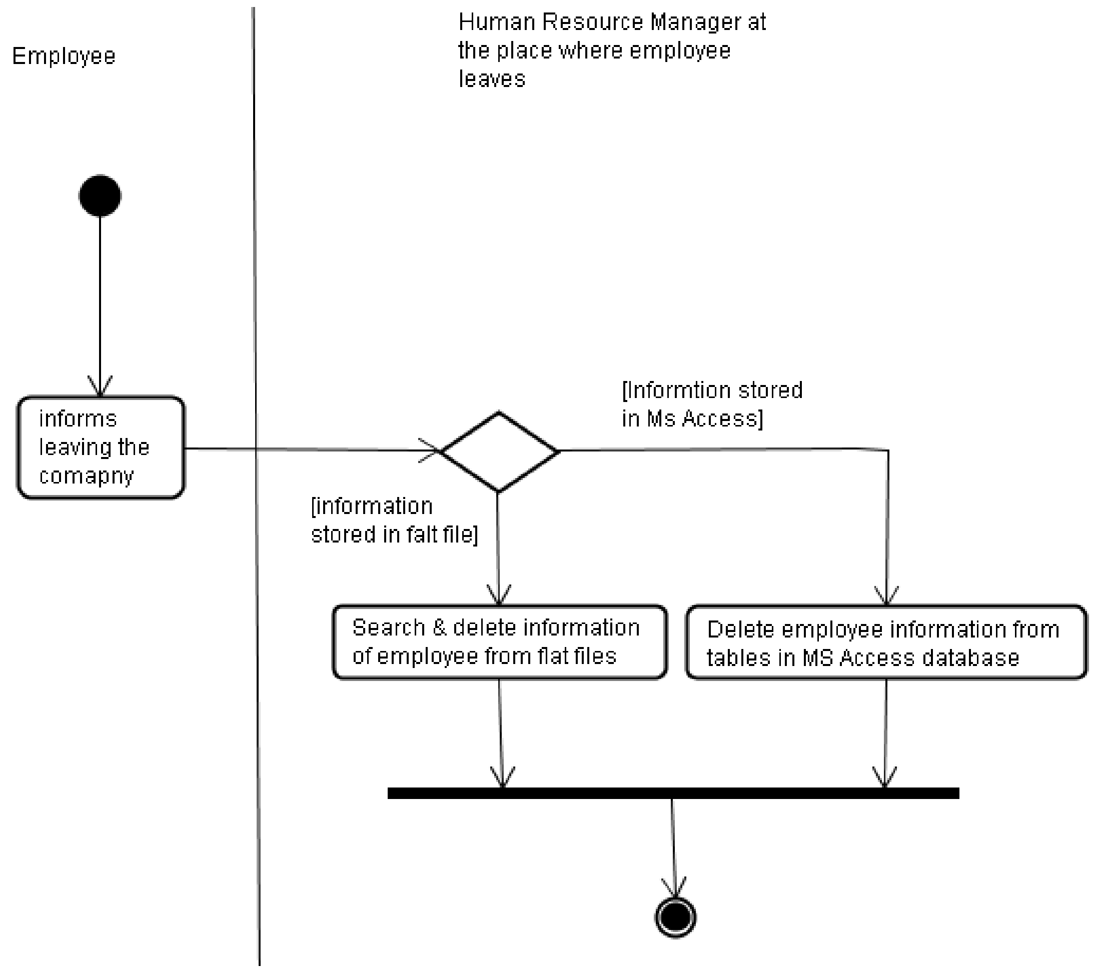

Employee
Human Resource Manager at the place where employee leaves
informs leaving the comapny
[Informtion stored in Ms Access]
[information stored in falt file]
Search & delete information of employee from flat files
Delete employee information from tables in MS Access database

Figure 16: Activity Diagram 7(Existing System)

3.1.2 Semi Structured Interview

Then I wanted to further investigate my observations in the sense I wanted to find out reasons for my observation. By the time I wanted to have an understanding whether the solution I am going to develop will suit them. So I decided to do a semi-structured interview with the five human resource managers.

I have included the interview plan in ***Appendix 1***.

After doing the semi-structured interview I came to following decisions according to the feedback I got from the human resource managers.

- All the five people have used Internet & very much familiar with Internet. So the proposed system accessing through a web browser will not be any problem to them.
- Four people have used Internet based mail systems such as GMail, Hotmail, and Yahoo Mail. So logging in to the system will not be any problem to them.
- All of them are not satisfied with the current report generating system.
- Four of them stated that the security given to the employee information is not enough at the moment.
- The sorted or filtered reports about the employees satisfying certain criteria within one place, within two/three places have to be created frequently. Reports considering all five places also have to be created.
- All of them wanted the report generation to be automated. They want reports to be generated just by one or two clicks.
- All of them stated that current way of storing the data in to the system & managing employee data is not efficient & time consuming.
- Three of them suggested that interaction with the employees should be faster than the current settings.

3.1.3 Questionnaire

Then I wanted to find out the employee's view of the existing system & their expectations from the new system. Since the number of employees is high I decided to give a questionnaire to selected employees from all five places in a way that employees were distributed within all departments.

Please be kind enough to refer ***Appendix 2*** for questionnaire.

Here below I am mentioning the decisions I took according to the feedback I got from the questionnaire.

- 96% of the employees are familiar with computers. 92% of the employees have used Internet before. So viewing the proposed system using a web browser will not be a problem to majority employees. Anyhow some kind of training should be provided to the remaining 4% who are not familiar with the computers.
- 90% of employees have used Internet based mail systems. So logging to the system will not be a problem to majority of the employees.
- None of the employees are satisfied with the current functioning of Human Resource Management System.
- All employees find getting their data, informing a change to the manager is extremely inefficient & time consuming.
- All of the employees need an efficient automated mechanism to be implemented in the human resource departments.

3.2 Synthesis

3.2.1Requirements Modeling

After having an understanding of the existing problem and its context I accounted an initial set of requirements at first. I used a short version of Volere template to list them. I had to further refine the requirements as I was continuing. I am showing the final version below. I found a shortened version of a Volare temple in an OU course book. I got the help of it as well for my work at that stage.

Reference:
M363 *Software engineering with objects* (2008) Unit 4., The Open University. p. 19-20

Human Resource Employee Management System user requirements

Product constraints

1. *Purpose of the product.* We have an organization for manufacturing computer hard disk drives. Its manufacturing plant is in Kandy & it has got another four sales offices within Sri Lanka. We as the human resource mangers have decided to develop a new computer system to generate reports about employees satisfying certain criteria, to manage the employee information efficiently, to allow the employee interact with their details at the human resource departments.
At present in four places employee information is stored in flat files. In the report generation at first human resource manager go through the flat file & types the report using Microsoft Word. Employee Interaction with their information is done manually through filling forms.
In the other place employee information is stored in the Microsoft Access 97. At that place human resource manager queries the Access database & create the report in Microsoft Word by typing. There also employee interaction is done manually.
In all places there isn’t an automated way to get reports considering the employees at other location or all five locations.

The new system should supply an efficient report generating functionalities, efficient way to manage employee information, Further more it should allow employees to interact with their information efficiently.

2. *Customer & Other Stakeholders*: Customers are the human resource department managers in the organization. The main stakeholders are employees, human resource managers, and heads of other departments.

3. *Users of the product:* The users of the system are employees & human resource managers. 95% of the users are familiar with the Internet. Other 5% also should be able to use the system without getting help from others.

-

4. *Requirement Constraints:* A working or workable solution has to be provided within six months. The system will be made accessible through a web browser.

5. *Naming Conventions & definitions:* None at present

6. *Relevant facts:* None at present.

7. *Assumptions*: None at present

Functional requirements

8. *Scope of the product* .The new system will be operational in all the five places manufacturing plant & four sales places. The system's interface will have to tackle with:

- human resource managers creating reports of employee details at their places, other places, within all five places satisfying a certain criteria

- human resource managers adding new employee records or removing employee records or changing employee records

- employees seeing the records about them selves

- employees reporting the changes, errors according to their views about their information.

9. *Functional Requirements.* The system shall:

- handle the employee data at all five places.

- generate the sorted or filtered reports about employees satisfying certain criteria within one place, two/three places, within all five places.

- allow recording the details of a newly joined employee.

- allow deleting the details of a employee leaving the company

- allow changing the employee details.

- allow employees to view their details.

- allow employees to inform human resource managers about the changes they think should occur in their details or errors in their records according to their view.

- view reported errors & changes about employee's personal data

Non-functional requirements

10. *Look-and-feel requirements*
LF 1: All the system's user interfaces should comply with a one particular format.

11. *Usability requirements*
U1: The system shall be easy to use with the help of a computer in an intranet.

12. *Performance requirements*
P 1: The system shall be able to respond simple user requests within 2 seconds.
P2: The system shall be able to respond complex user requests within 15 seconds.
P3: The system shall be available within all the office hours.

13. *Operational requirements*
O1: The system shall be accessible through any computer in an intranet between five places.
O2: The system shall be host at a single central server.
.
14. *Security requirements*
S1: Only human resource managers should be able to generate employee reports, add, delete, and change employee records, view reported errors & changes.
S2: Employees shall be able only to view their own information & inform human resource managers if they find that something in their information should be changed or contains errors.

After that I thought about the proposed software solution in terms of users view & started developing the use case diagram for the proposed system.

Use case Diagram: I drew an initial use case diagram at first. Then when I was going further I did some changes in the in the first version & created a second version of the use case diagram. The first version I am including in the ***Appendix 3***. The second version I am mentioning below.

-

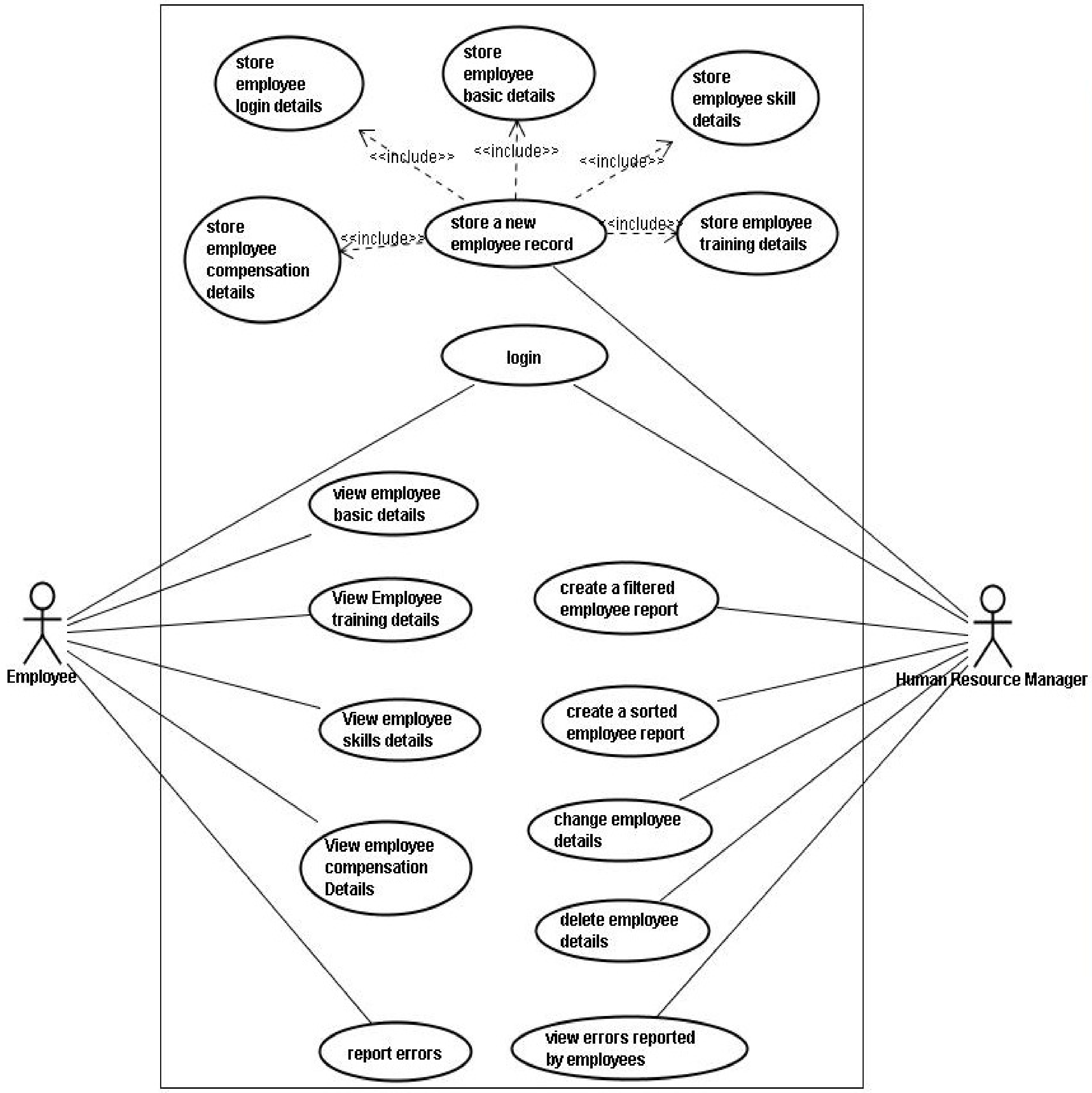

store employee login details
store employee basic details
store employee skill details
<<include>>
<<include>>
<<include>>
store employee compensation details
<<include>>
store a new employee record
<<include>>
store employee training details
login
view employee basic details
View Employee training details
View employee skills details
View employee compensation Details
report errors
create a filtered employee report
create a sorted employee report
change employee details
delete employee details
view errors reported by employees
Employee
Human Resource Manager

Figure 17: Final Use Case Diagram for the proposed solution

Then I wanted to further elaborate these use cases in the above diagram & created textual descriptions of the use cases.

<u>**Table 1** A textual description for *login* use case</u>

Identifier and name	UC1 *login*
Initiator	*Human Resource Manager or Employee*
Goal	Human resource manager or employee is logged in to the system.
Precondition	Human resource manager or employee has a user name & a password to log on to the system.
Post condition	Human resource manager or employee will be logged in to the system. They will be forwarded to their home pages.
Assumptions	The expected initiator is a Human Resource manager or existing employee, using a web browser to perform the use case.

Main Success Scenario

1. The Human resource employee management system requests a user name & a password.
2. The employee or human resource manager provides user name & password.
3. The human resource management system checks whether the user name & password combination is valid.
4. If the user name & password combination is valid human resource manager or employee will be directed to their home pages.

Extensions

4.a. Invalid username & password combination

4.a.1 An error message will be displayed saying “Invalid username or password”.

4.a.2 The scenario continues at step 1.

Table 2 A textual description for *store a new employee record* use case

Identifier and name	UC2 *enter new employee record*
Initiator	*Human Resource Manager*
Goal	A new record is inserted to appropriate tables for an employee
Precondition	Human resource manager is logged in to the system.
Post condition	A new record will be inserted to each table regarding employee Basic details, Compensation details, and Skills details. New user name & password will be created for each employee & stored in the table corresponding to login information. Each employee will be assigned to employee group & stored in table corresponding to group information. If the employee is a trainee a record will be entered to the Training table as well.
Assumptions	The expected initiator is a Human Resource manager, using a web browser to perform the use case.

Main Success Scenario

1. The human resource manager makes a request to enter new employee data.
2. The human resource employee management system requests whether the employee is a trainee or not.
3. The human resource manager confirms the fact trainee or not.
4. The human resource manager enters the employee details & a proposed new user name & password.
5. The human resource employee management system enters the data to corresponding tables in the database.

Extensions

None

Table 3 A textual description for *create a filtered employee report*

Identifier and name	UC3 *view filtered employee data report*
Initiator	*Human Resource Manager*
Goal	A printable report satisfying the supplied criteria is generated.
Precondition	Human Resource manager is logged in to the system.
Post condition	A pintable report containing the information of employee/s filtered according to the given criteria will be generated.
Assumptions	The expected initiator is a human resource manager, using a web browser to perform the use case.

Main Success Scenario

1. The human resource manager makes a request to create a filtered employee data report.
2. The human resource employee management system provides the available filter options.
3. The human resource manager selects the desired option.
4. The system requests filter criteria.
5. The human resource manager enters the filter criteria.
6. The system creates the filtered employee data report.

Extensions

None

Table 4 A textual description for *create a sorted employee report*

Identifier and name	UC4 *create a sorted employee report*
Initiator	*Human Resource Manager*
Goal	A printable report satisfying the sorting criteria is generated.
Precondition	Human resource manager is logged in to the system.
Post condition	A printable report containing the information of employee/s sorted according to the given sorting criteria will be generated.
Assumptions	The expected initiator is a human resource manager, using a web browser to perform the use case.

Main Success Scenario

1. The human resource manager makes a request to create a sorted employee data report.
2. The human resource employee management system provides the available sorting options.
3. The human resource manager selects the desired option.
4. The system requests sorting criteria.
5. The human resource manager enters the sorting criteria.
6. The system creates the sorted employee data report.

Extensions

None

<u>**Table 5** A textual description for *change employee details*</u>

Identifier and name	UC5 *change employee details*
Initiator	*Human Resource Manager*
Goal	The selected employee details are updated with supplied values.
Precondition	Human Resource manager is logged in to the system.
Post condition	The selected employee details will be updated by supplied values.
Assumptions	The expected initiator is a human resource manager, using a web browser to perform the use case.

Main Success Scenario

1. The human resource manager makes a request to change the employee data.
2. The human resource employee management system requests the employee ID of the employee whose details has to be changed.
3. The human resource manager enters the employee ID of the employee whose details has to be changed.
4. The system provides the data fields that can be updated.
5. The human resource manager selects the desired data field.
6. The system requests new value.
7. The human resource manager enters the new value.
8. The system updates the selected data field with the new value

Extensions

None

-

Table 6 A textual description for *delete employee details*

Identifier and name	UC6 *delete employee details*
Initiator	*Human Resource Manager*
Goal	The selected employee's details are deleted from selected tables.
Precondition	Human Resource manager is logged in to the system.
Post condition	The selected employee's details will be deleted from selected tables.
Assumptions	The expected initiator is a human resource manager, using a web browser to perform the use case.

Main Success Scenario

1. The human resource manager makes a request to delete the employee data.
2. The human resource employee management system requests the employee ID of the employee whose details has to be deleted.
3. The human resource manager enters the employee ID of the employee whose details has to be deleted.
4. The system provides the table names from which data can be deleted.
5. The human resource manager selects the desired table names.
6. The system deletes the data of the employee having selected ID from selected tables.

Extensions

None

Table 7 A textual description for *report errors*

Identifier and name	UC7 *report errors*
Initiator	*Employee*
Goal	The selected employee's error message is stored in the system.
Precondition	Employee is logged in to the system.
Post condition	The employee's error message will be stored in the human resource management system.
Assumptions	The expected initiator is an existing employee of the company, using a web browser to perform the use case.

Main Success Scenario

1. The employee makes a request to report errors.
2. The human resource employee management system requests the error message.
3. The employee types the error.
4. The system stores the error message

Extensions

None

Table 8 A textual description for *view errors reported by employees*

Identifier and name	UC8 *view errors reported by employees*
Initiator	*Human Resource Manager*
Goal	Errors reported by employees about their details are displayed to human resource manager.
Precondition	Human resource manager is logged in to the system.
Post conditions	Errors reported by employees about their details are displayed to human resource manager in tabular format.
Assumptions	The expected initiator is a Human Resource manager, using a web browser to perform the use case.

Main Success Scenario

1. The employee makes a request to view errors reported by employees.
2. The system displays the details about errors reported by employees in a tabular format.

Extensions

None

Table 9 A textual description for *view employee basic details*

Identifier and name	UC9 *view employee basic details*
Initiator	*Employee*
Goal	Basic details of the initiator (an employee) is displayed to him/her self.
Precondition	Employee is logged in to the system.
Post condition	Basic details of the initiator is displayed to him/her self in tabular format.
Assumptions	The expected initiator is an existing employee of the company, using a web browser to perform the use case.

Main Success Scenario

1. The employee makes a request to view basic details of him/her self.
2. The system displays the basic details of the employee in a tabular format

Extensions

None

Table 10 A textual description for *view employee skills details*

Identifier and name	UC10 *view employee skill details*
Initiator	*Employee*
Goal	Skills details of the initiator (an employee) are displayed to him/her self.
Precondition	Employee is logged in to the system.
Post condition	Skills details of the initiator is displayed to him/her self in tabular format.

Assumptions	The expected initiator is an existing employee of the company, using a web browser to perform the use case.

Main Success Scenario
1.The employee makes a request to view skills details of him/her self.
2. The system displays the skills details of the employee in a tabular format

Table 11 A textual description for *view employee compensation details*

Identifier and name	UC11 *view employee compensation details*
Initiator	*Employee*
Goal	Compensation details of the initiator (an employee) are displayed to him/her self.
Precondition	Employee is logged in to the system.
Post condition	Compensation details of the initiator is displayed to him/her self in tabular format.
Assumptions	The expected initiator is an existing employee of the company, using a web browser to perform the use case.

Main Success Scenario
1. The employee makes a request to view compensation details of him/her self.
2. The system displays the compensation details of the employee in a tabular format.

Extensions

None

Table 12 A textual description for *view employee training details*

Identifier and name	UC12 *view employee training details*
Initiator	*Employee*
Goal	Training details of the initiator (an employee) are displayed to him/her self.
Precondition	Employee is logged in to the system. He/She is a trainee employee.
Post condition	Training details of the initiator is displayed to him/her self in tabular format.

Assumptions	The expected initiator is an existing employee of the company, using a web browser to perform the use case.

Main Success Scenario

1. The employee makes a request to view training details of him/her self.
2. The system displays the Training details of the employee in a tabular format.

Extensions

None

3.2.2 Static & dynamic modeling

After that I again went through the requirement documentation, use cases, their textual descriptions started modeling the static view of the proposed prototype software solution.

I went through the requirements and identified the needed boundary classes, control classes as well as entity classes. Then I draw the class diagram for my solution. In the mid of the project I had to review my class diagram & go for another iteration of the diagram since a new requirement evolved when I was continuing 'Employee should be able to report human resource managers if he/she finds errors in their records'. Here I am mentioning below the final class diagram & I'll put the first version in to the ***Appendix 4***.

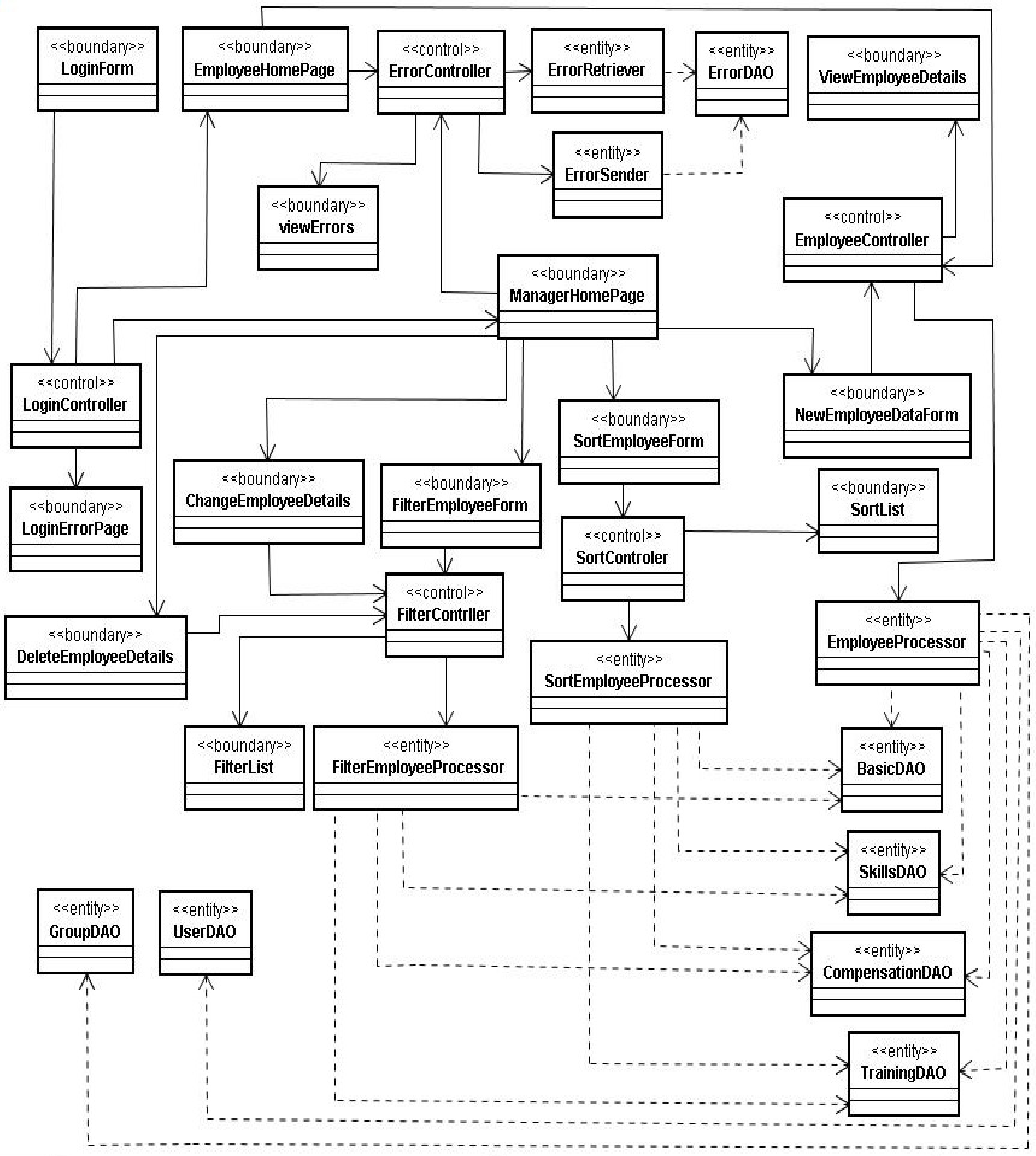

<<boundary>> LoginForm
<<boundary>> EmployeeHomePage
<<control>> ErrorController
<<entity>> ErrorRetriever
<<entity>> ErrorDAO
<<boundary>> ViewEmployeeDetails
<<entity>> ErrorSender
<<boundary>> viewErrors
<<control>> EmployeeController
<<boundary>> ManagerHomePage
<<control>> LoginController
<<boundary>> NewEmployeeDataForm
<<boundary>> SortEmployeeForm
<<boundary>> LoginErrorPage
<<boundary>> ChangeEmployeeDetails
<<boundary>> FilterEmployeeForm
<<boundary>> SortList
<<control>> SortControler
<<control>> FilterContrller
<<entity>> EmployeeProcessor
<<boundary>> DeleteEmployeeDetails
<<entity>> SortEmployeeProcessor
<<boundary>> FilterList
<<entity>> FilterEmployeeProcessor
<<entity>> BasicDAO
<<entity>> SkillsDAO
<<entity>> GroupDAO
<<entity>> UserDAO
<<entity>> CompensationDAO
<<entity>> TrainingDAO

Figure 18: Final Class Diagram for the proposed solution

-

I modeled the dynamic behavior of the system. I went through the use cases, class diagrams & decided how messages were flowing between the objects of my proposed class diagram. I developed sequence diagrams for all most all the use cases. Please be kind enough to magnify the diagrams & see. I tried my best to make it clear.

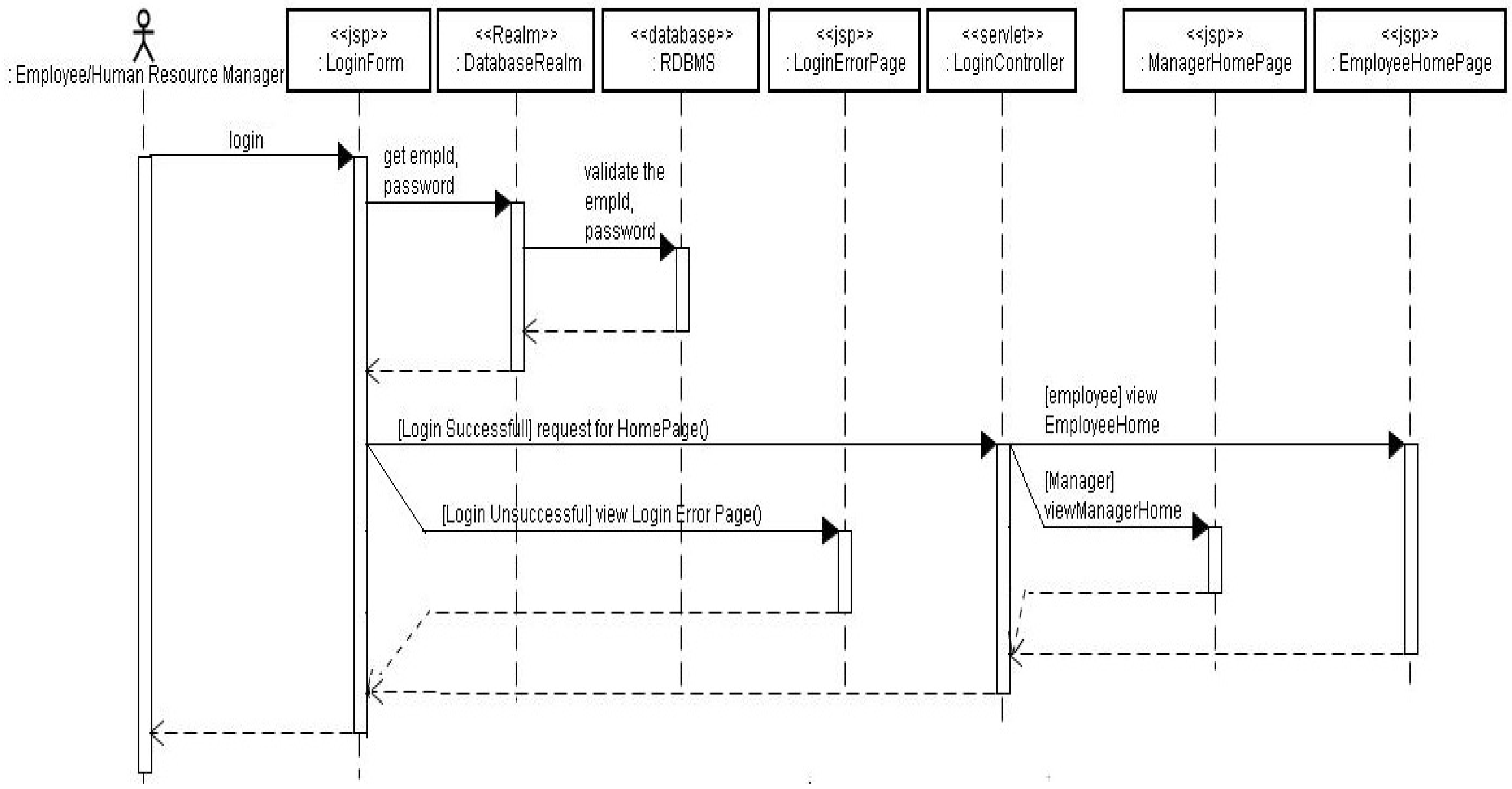

: Employee/Human Resource Manager
<<jsp>> : LoginForm
<<Realm>> : DatabaseRealm
<<database>> : RDBMS
<<jsp>> : LoginErrorPage
<<servlet>> : LoginController
<<jsp>> : ManagerHomePage
<<jsp>> : EmployeeHomePage
login
get empId, password
validate the empId, password
[Login Successfull] request for HomePage()
[Login Unsuccessful] view Login Error Page()
[employee] view EmployeeHome
[Manager] viewManagerHome

Figure 19: login sequence diagram

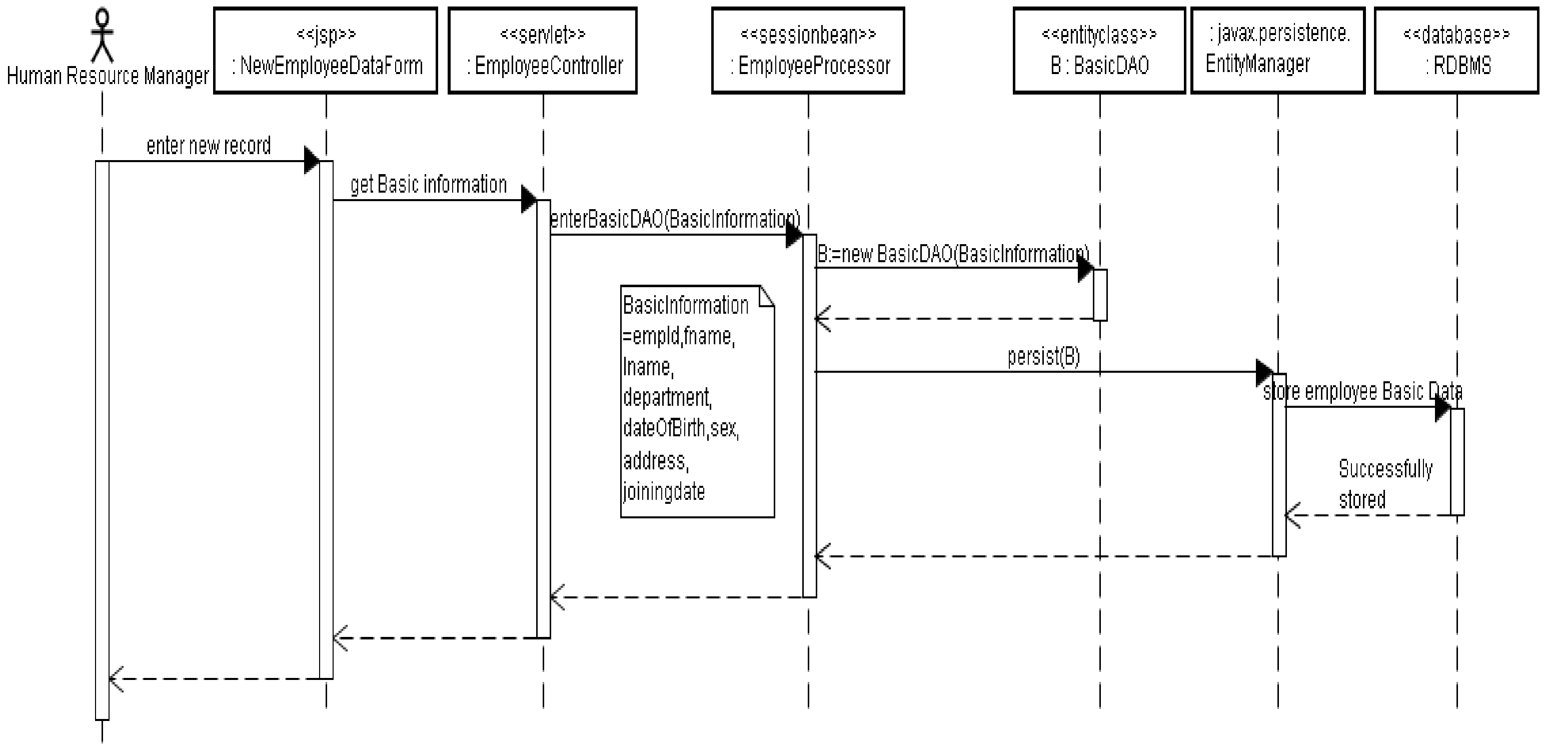


Figure 20: store employee basic details sequence diagram

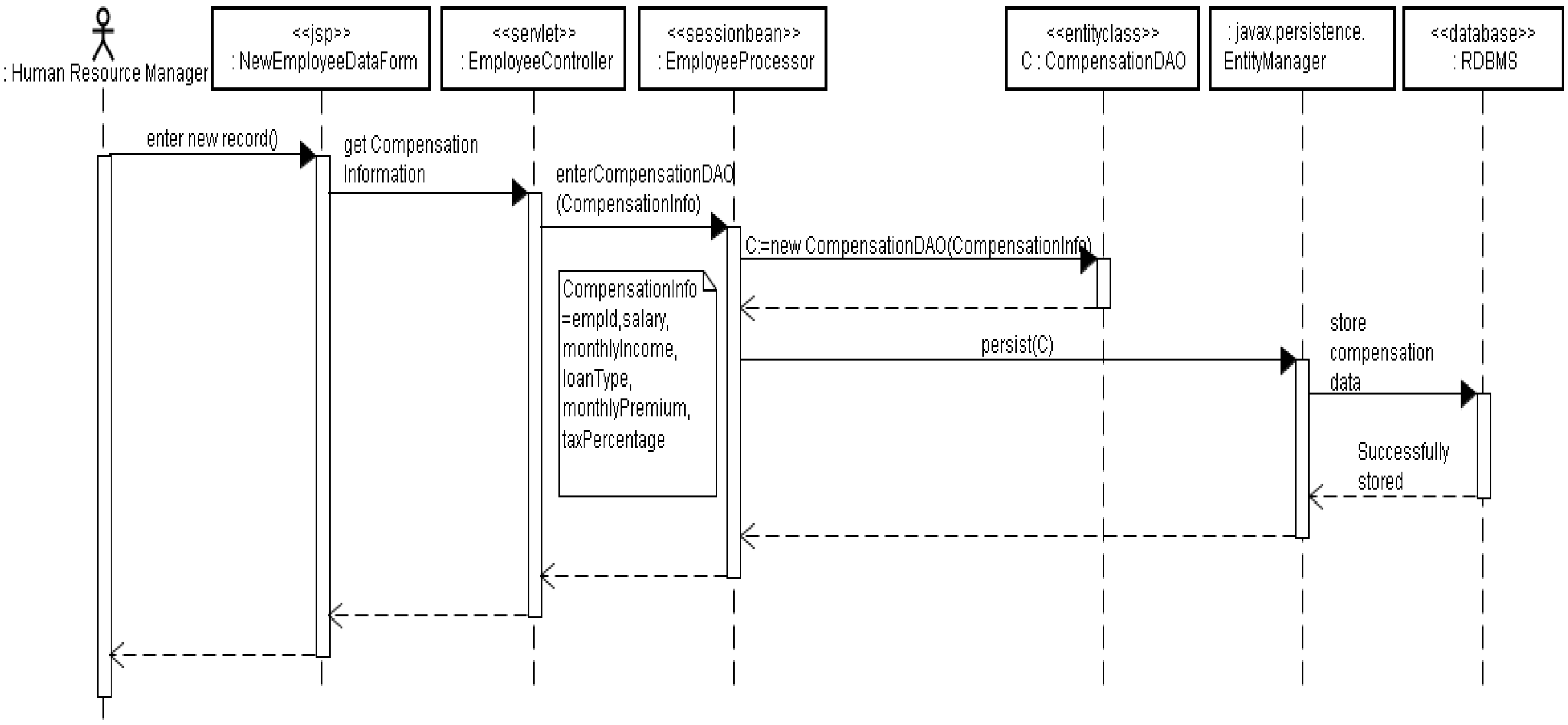


Figure 21: store employee compensation details sequence diagram

-

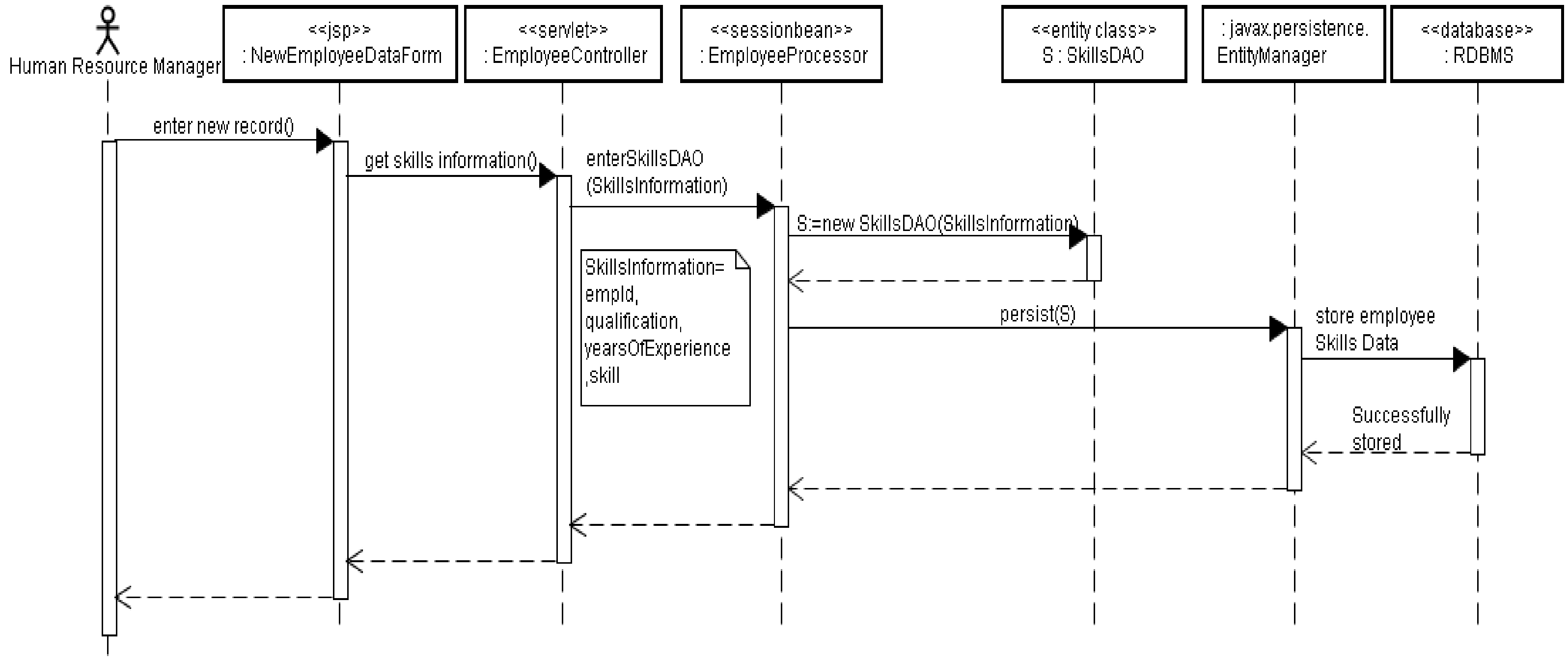

Human Resource Manager
<<jsp>> : NewEmployeeDataForm
<<servlet>> : EmployeeController
<<sessionbean>> : EmployeeProcessor
<<entity class>> S : SkillsDAO
: javax.persistence. EntityManager
<<database>> : RDBMS
enter new record()
get skills information()
enterSkillsDAO (SkillsInformation)
S:=new SkillsDAO(SkillsInformation)
SkillsInformation= empId, qualification, yearsOfExperience ,skill
persist(S)
store employee Skills Data
Successfully stored

Figure 22: store employee skills details sequence diagram

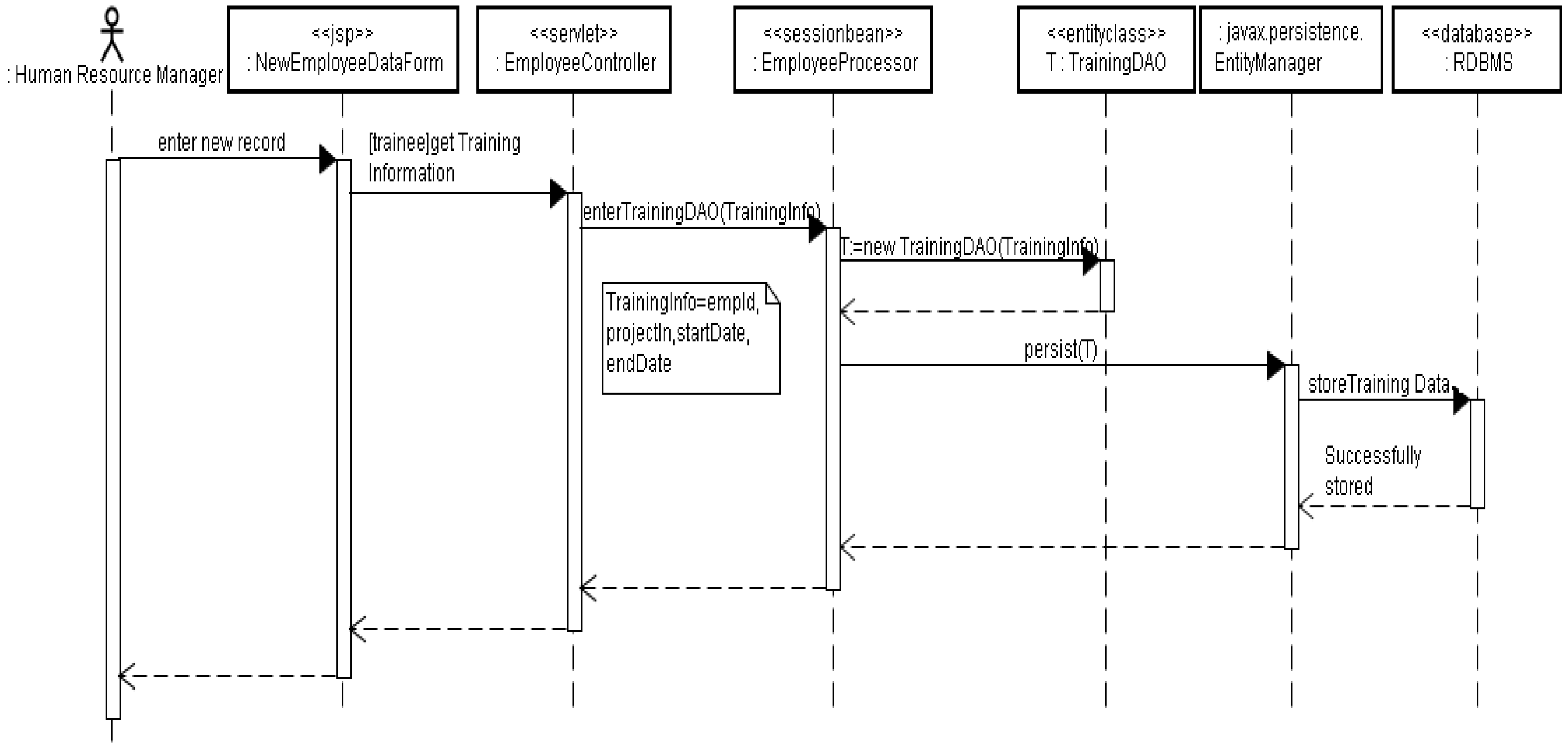

: Human Resource Manager
<<jsp>> : NewEmployeeDataForm
<<servlet>> : EmployeeController
<<sessionbean>> : EmployeeProcessor
<<entityclass>> T : TrainingDAO
: javax.persistence. EntityManager
<<database>> : RDBMS
enter new record
[trainee]get Training Information
enterTrainingDAO(TrainingInfo)
T:=new TrainingDAO(TrainingInfo)
TrainingInfo=empId, projectIn,startDate, endDate
persist(T)
storeTraining Data
Successfully stored

Figure 23: store employee training details sequence diagram

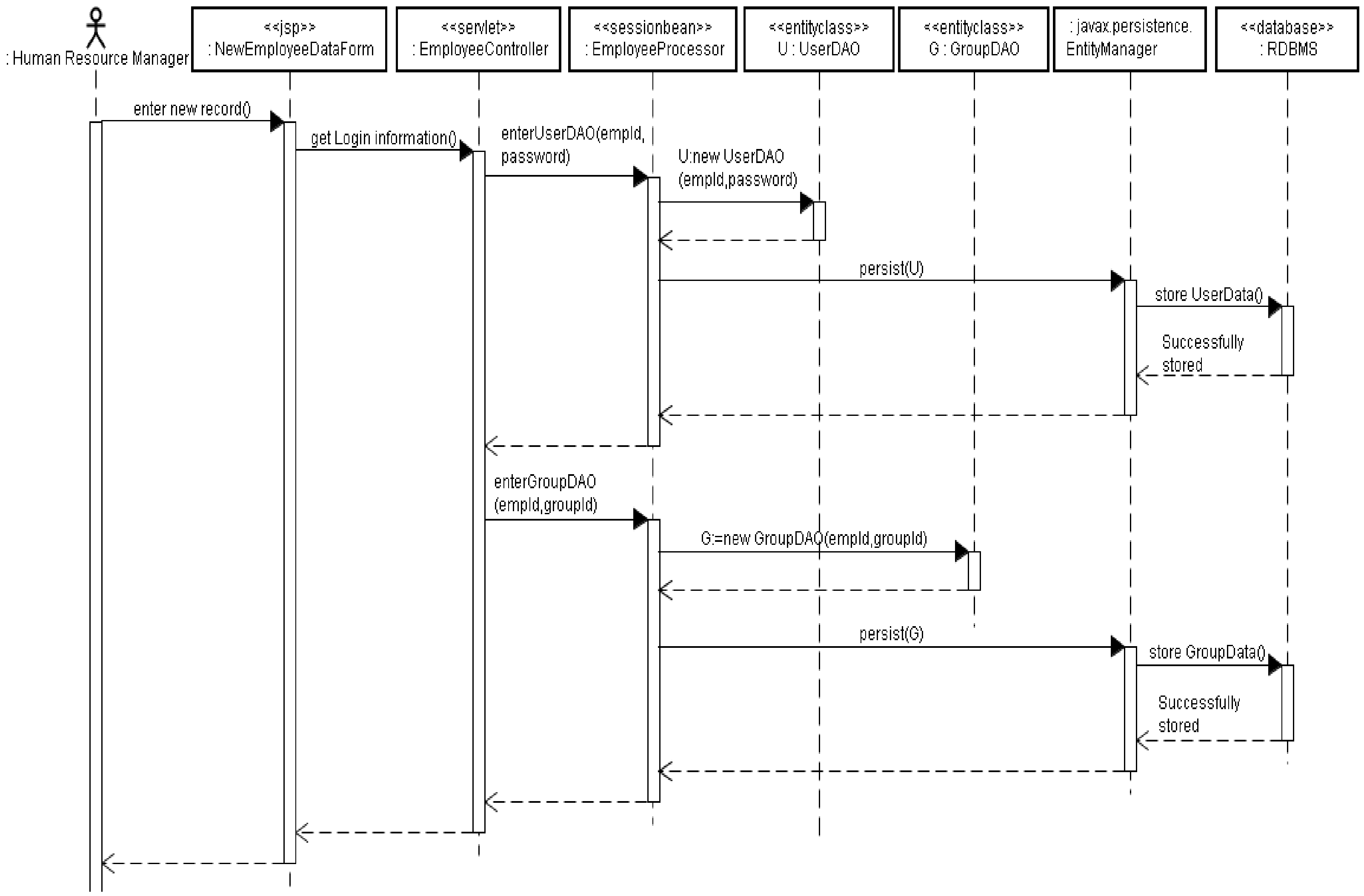

: Human Resource Manager
<<jsp>> : NewEmployeeDataForm
<<servlet>> : EmployeeController
<<sessionbean>> : EmployeeProcessor
<<entityclass>> U : UserDAO
<<entityclass>> G : GroupDAO
: javax.persistence. EntityManager
<<database>> : RDBMS
enter new record()
get Login information()
enterUserDAO(empId, password)
U:new UserDAO (empId,password)
persist(U)
store UserData()
Successfully stored
enterGroupDAO (empId,groupId)
G:=new GroupDAO(empId,groupId)
persist(G)
store GroupData()
Successfully stored

Figure 24: store employee login details

-

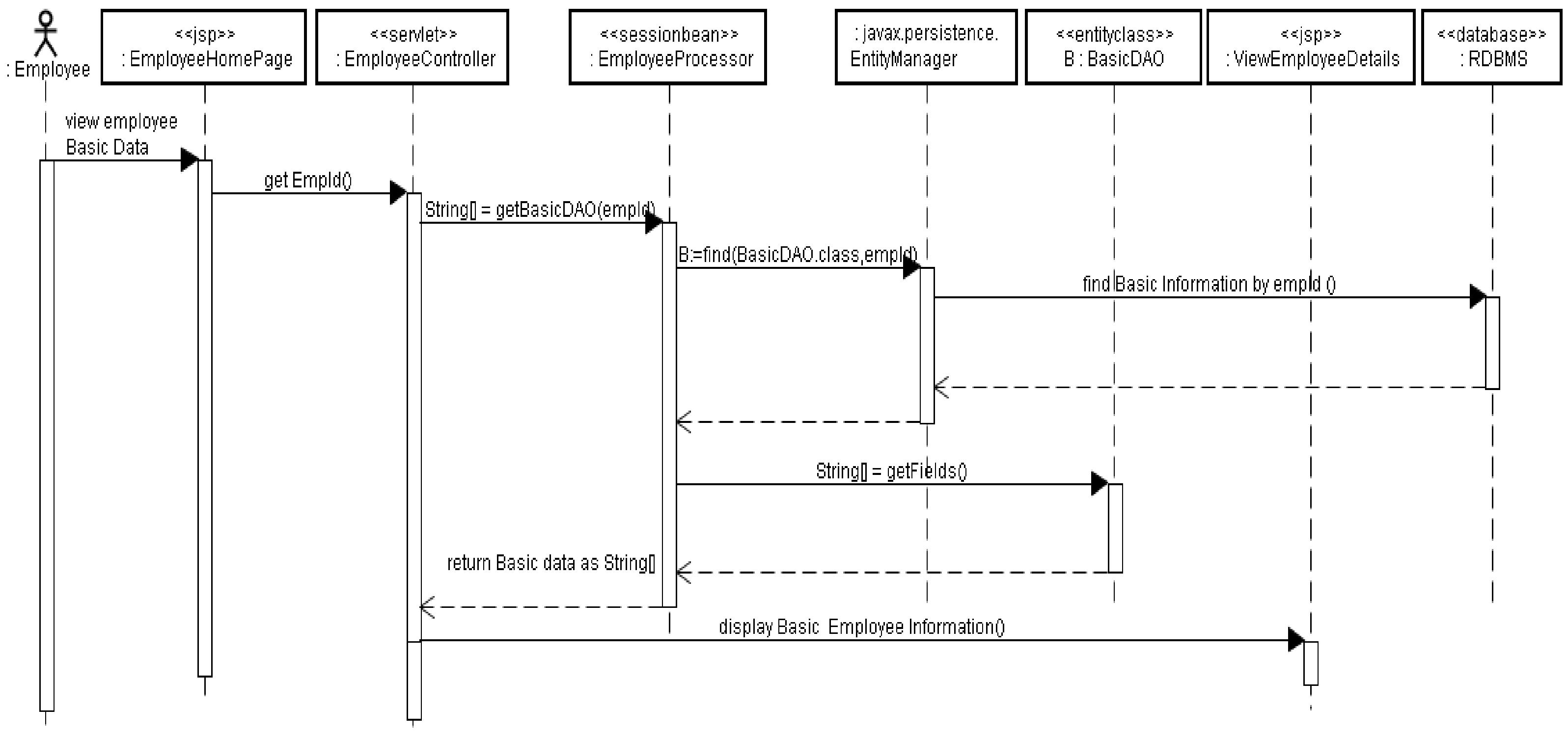

: Employee
<<jsp>> : EmployeeHomePage
<<servlet>> : EmployeeController
<<sessionbean>> : EmployeeProcessor
: javax.persistence. EntityManager
<<entityclass>> B : BasicDAO
<<jsp>> : ViewEmployeeDetails
<<database>> : RDBMS
view employee Basic Data
get EmpId()
String[] = getBasicDAO(empId)
B:=find(BasicDAO.class,empId)
find Basic Information by empId ()
String[] = getFields()
return Basic data as String[]
display Basic Employee Information()

Figure 25: view employee basic details sequence diagram

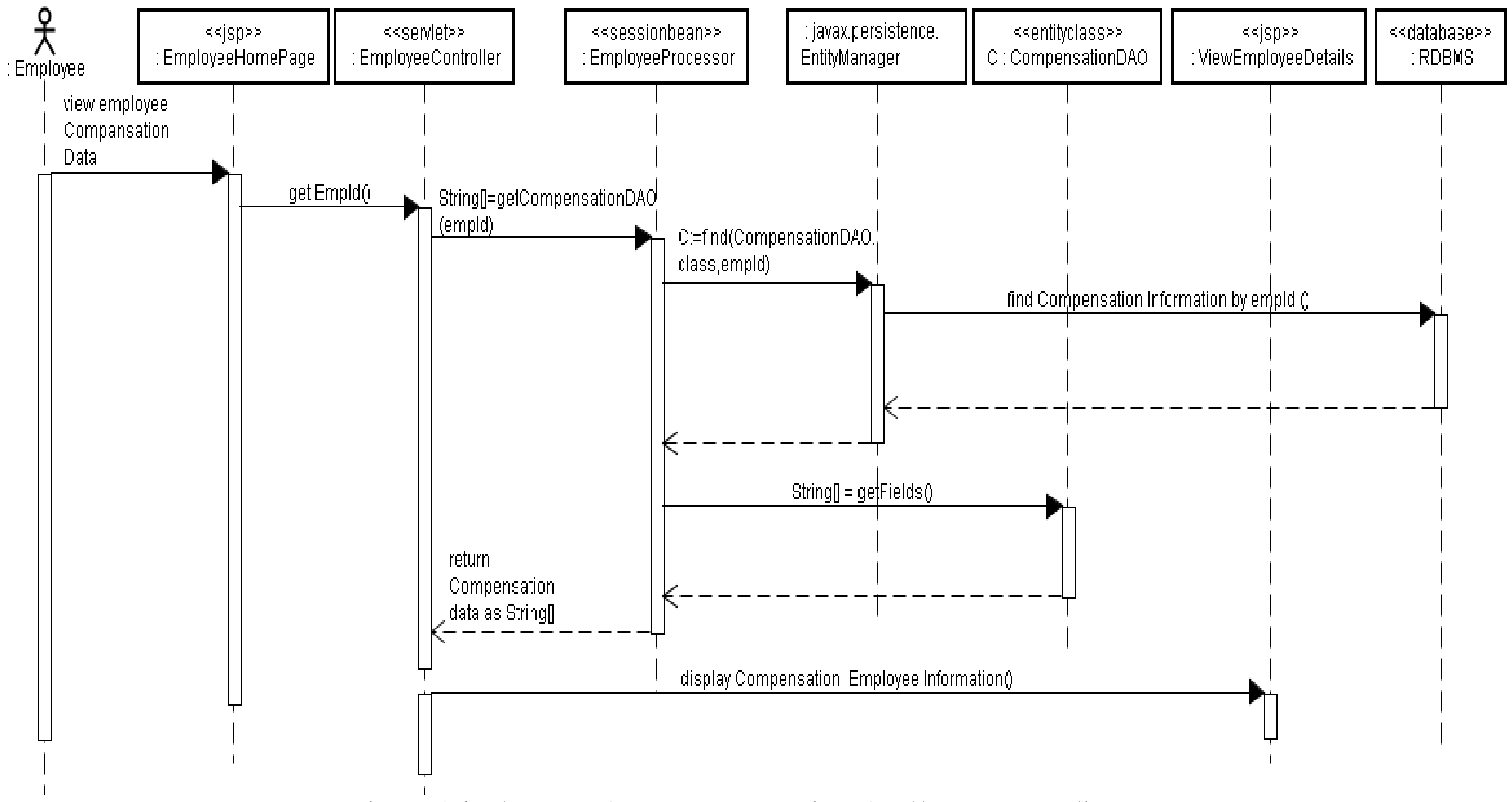

: Employee
<<jsp>> : EmployeeHomePage
<<servlet>> : EmployeeController
<<sessionbean>> : EmployeeProcessor
: javax.persistence. EntityManager
<<entityclass>> C : CompensationDAO
<<jsp>> : ViewEmployeeDetails
<<database>> : RDBMS
view employee Compansation Data
get EmpId()
String[]=getCompensationDAO (empId)
C:=find(CompensationDAO. class,empId)
find Compensation Information by empId ()
String[] = getFields()
return Compensation data as String[]
display Compensation Employee Information()

Figure 26: view employee compensation details sequence diagram

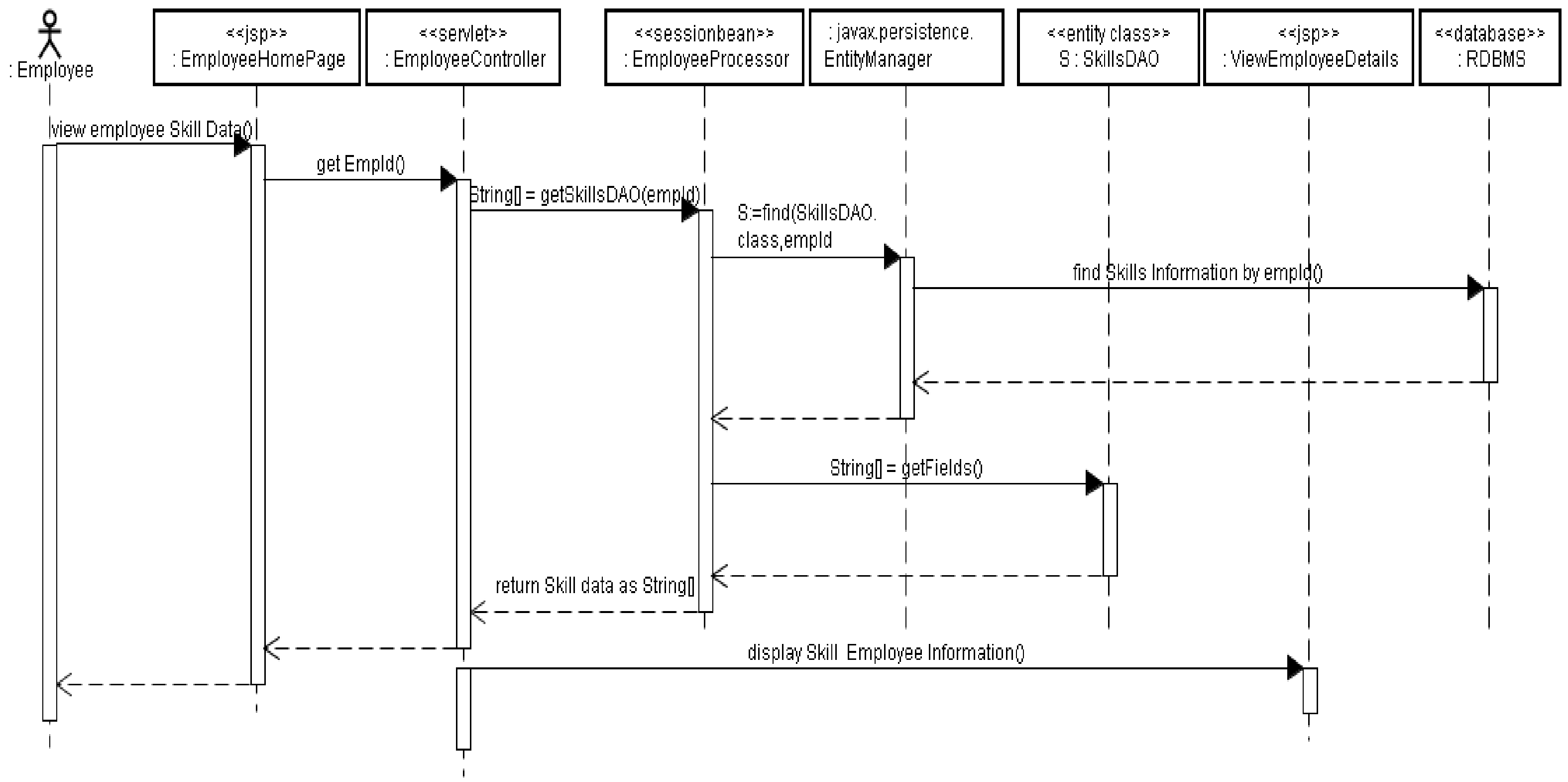

: Employee
<<jsp>> : EmployeeHomePage
<<servlet>> : EmployeeController
<<sessionbean>> : EmployeeProcessor
: javax.persistence. EntityManager
<<entity class>> S : SkillsDAO
<<jsp>> : ViewEmployeeDetails
<<database>> : RDBMS
view employee Skill Data()
get EmpId()
String[] = getSkillsDAO(empId)
S:=find(SkillsDAO. class,empId
find Skills Information by empId()
String[] = getFields()
return Skill data as String[]
display Skill Employee Information()

Figure 27: view employee skills details sequence diagram

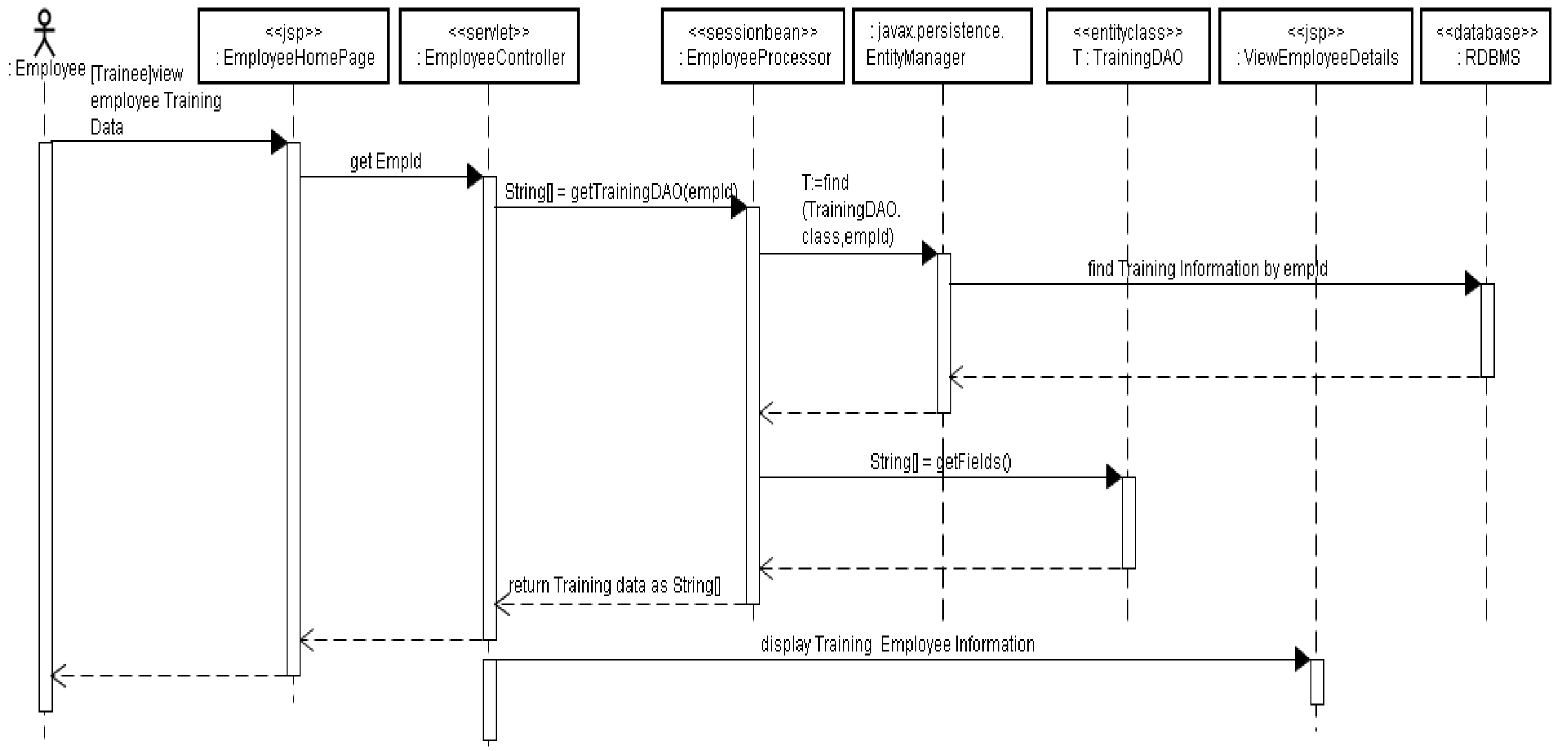

: Employee
<<jsp>> : EmployeeHomePage
<<servlet>> : EmployeeController
<<sessionbean>> : EmployeeProcessor
: javax.persistence. EntityManager
<<entityclass>> T : TrainingDAO
<<jsp>> : ViewEmployeeDetails
<<database>> : RDBMS
[Trainee]view employee Training Data
get EmpId
String[] = getTrainingDAO(empId)
T:=find (TrainingDAO. class,empId)
find Training Information by empId
String[] = getFields()
return Training data as String[]
display Training Employee Information

Figure 28: view employee training details sequence diagram

-

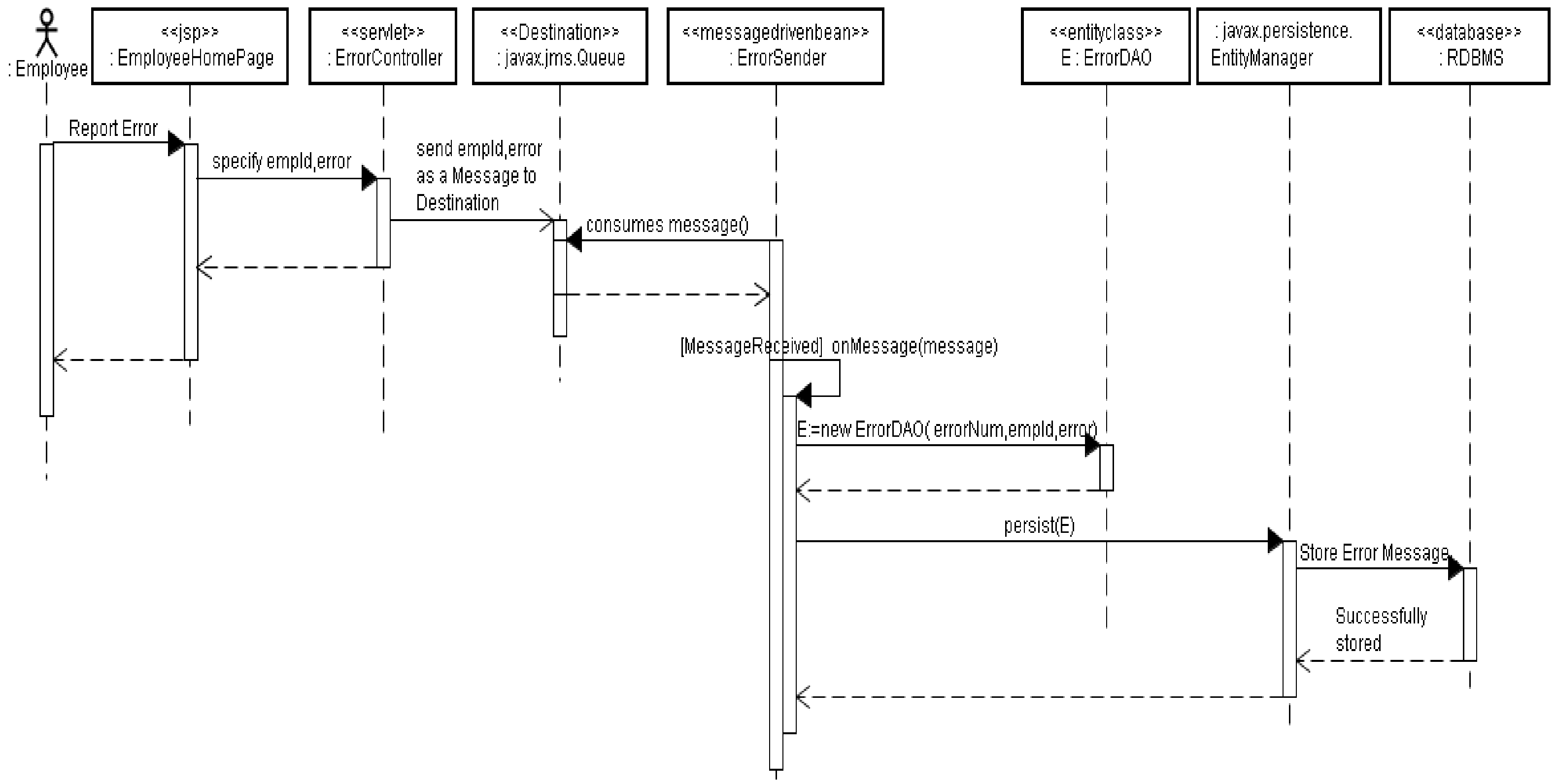

: Employee
<<jsp>> : EmployeeHomePage
<<servlet>> : ErrorController
<<Destination>> : javax.jms.Queue
<<messagedrivenbean>> : ErrorSender
<<entityclass>> E : ErrorDAO
: javax.persistence. EntityManager
<<database>> : RDBMS
Report Error
specify empId,error
send empId,error as a Message to Destination
consumes message()
[MessageReceived] onMessage(message)
E:=new ErrorDAO(errorNum,empId,error)
persist(E)
Store Error Message
Successfully stored

Figure 28: report errors sequence diagrams

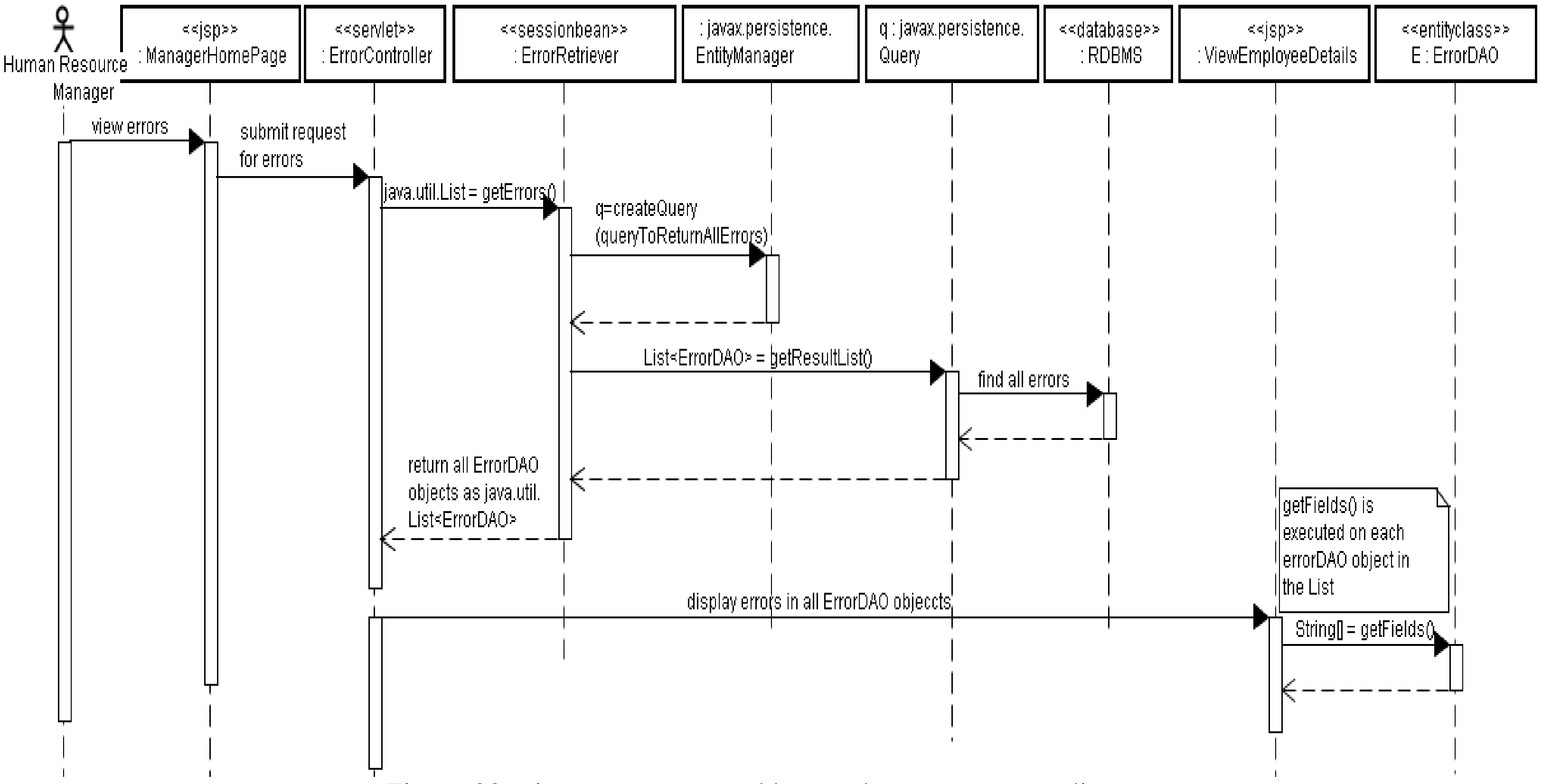

Human Resource Manager
<<jsp>> : ManagerHomePage
<<servlet>> : ErrorController
<<sessionbean>> : ErrorRetriever
: javax.persistence. EntityManager
q : javax.persistence. Query
<<database>> : RDBMS
<<jsp>> : ViewEmployeeDetails
<<entityclass>> E : ErrorDAO
view errors
submit request for errors
java.util.List = getErrors()
q=createQuery (queryToReturnAllErrors)
List<ErrorDAO> = getResultList()
find all errors
return all ErrorDAO objects as java.util. List<ErrorDAO>
display errors in all ErrorDAO objeccts
getFields() is executed on each errorDAO object in the List
String[] = getFields()

Figure 29: view errors reported by employees sequence diagram

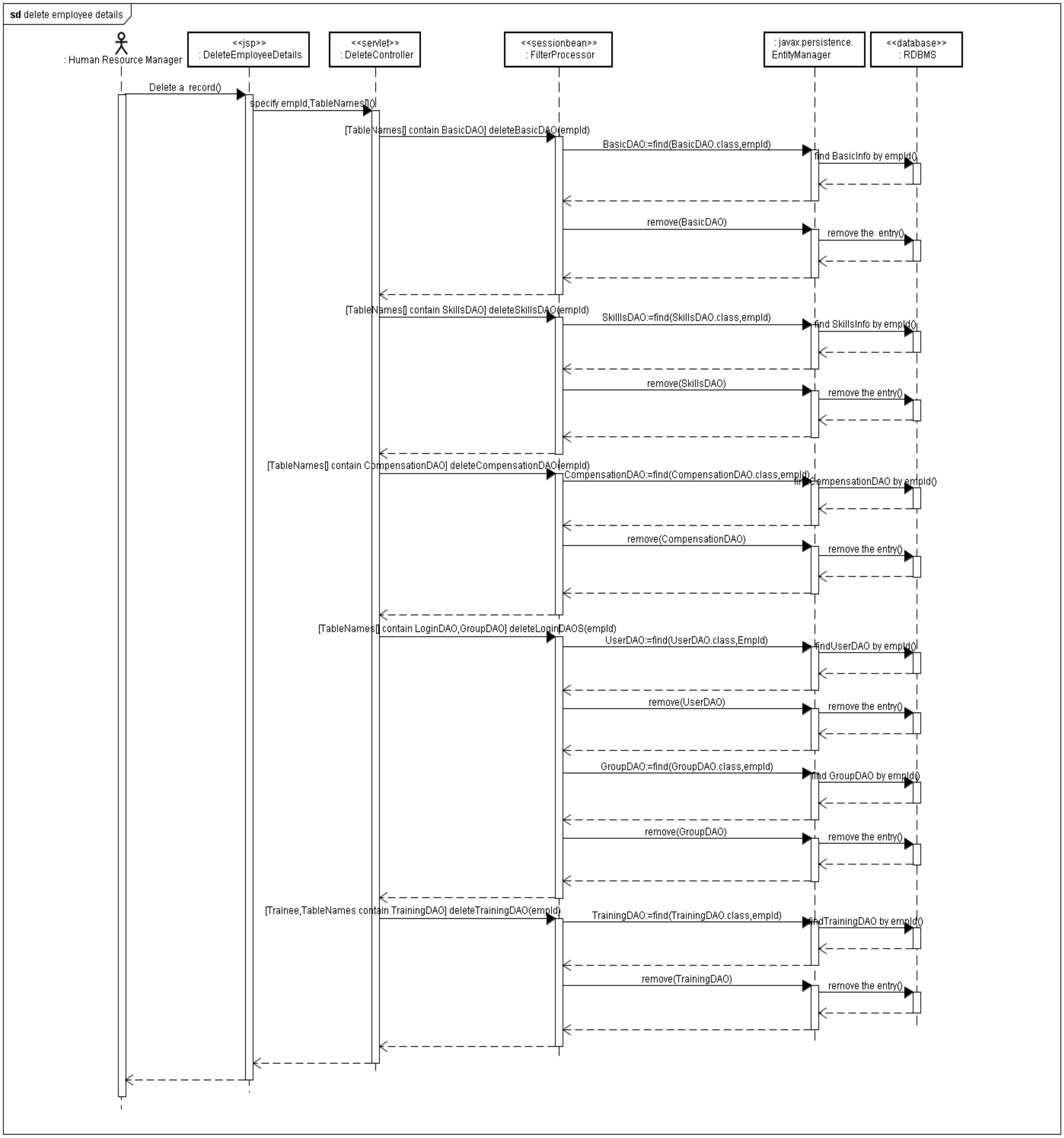

sd delete employee details
: Human Resource Manager
<<jsp>> : DeleteEmployeeDetails
<<servlet>> : DeleteController
<<sessionbean>> : FilterProcessor
: javax.persistence. EntityManager
<<database>> : RDBMS
Delete a record()
specify empId,TableNames[]()
[TableNames[] contain BasicDAO] deleteBasicDAO(empId)
BasicDAO:=find(BasicDAO.class,empId)
find BasicInfo by empId()
remove(BasicDAO)
remove the entry()
[TableNames[] contain SkillsDAO] deleteSkillsDAO(empId)
SkillsDAO:=find(SkillsDAO.class,empId)
find SkillsInfo by empId()
remove(SkillsDAO)
remove the entry()
[TableNames[] contain CompensationDAO] deleteCompensationDAO(empId)
CompensationDAO:=find(CompensationDAO.class,empId)
find CompensationDAO by empId()
remove(CompensationDAO)
remove the entry()
[TableNames[] contain LoginDAO,GroupDAO] deleteLoginDAOS(empId)
UserDAO:=find(UserDAO.class,EmpId)
findUserDAO by empId()
remove(UserDAO)
remove the entry()
GroupDAO:=find(GroupDAO.class,empId)
find GroupDAO by empId()
remove(GroupDAO)
remove the entry()
[Trainee,TableNames contain TrainingDAO] deleteTrainingDAO(empId)
TrainingDAO:=find(TrainingDAO.class,empId)
findTrainingDAO by empId()
remove(TrainingDAO)
remove the entry()

Figure 29: delete employee details sequence diagram(Please magnify & see).

-

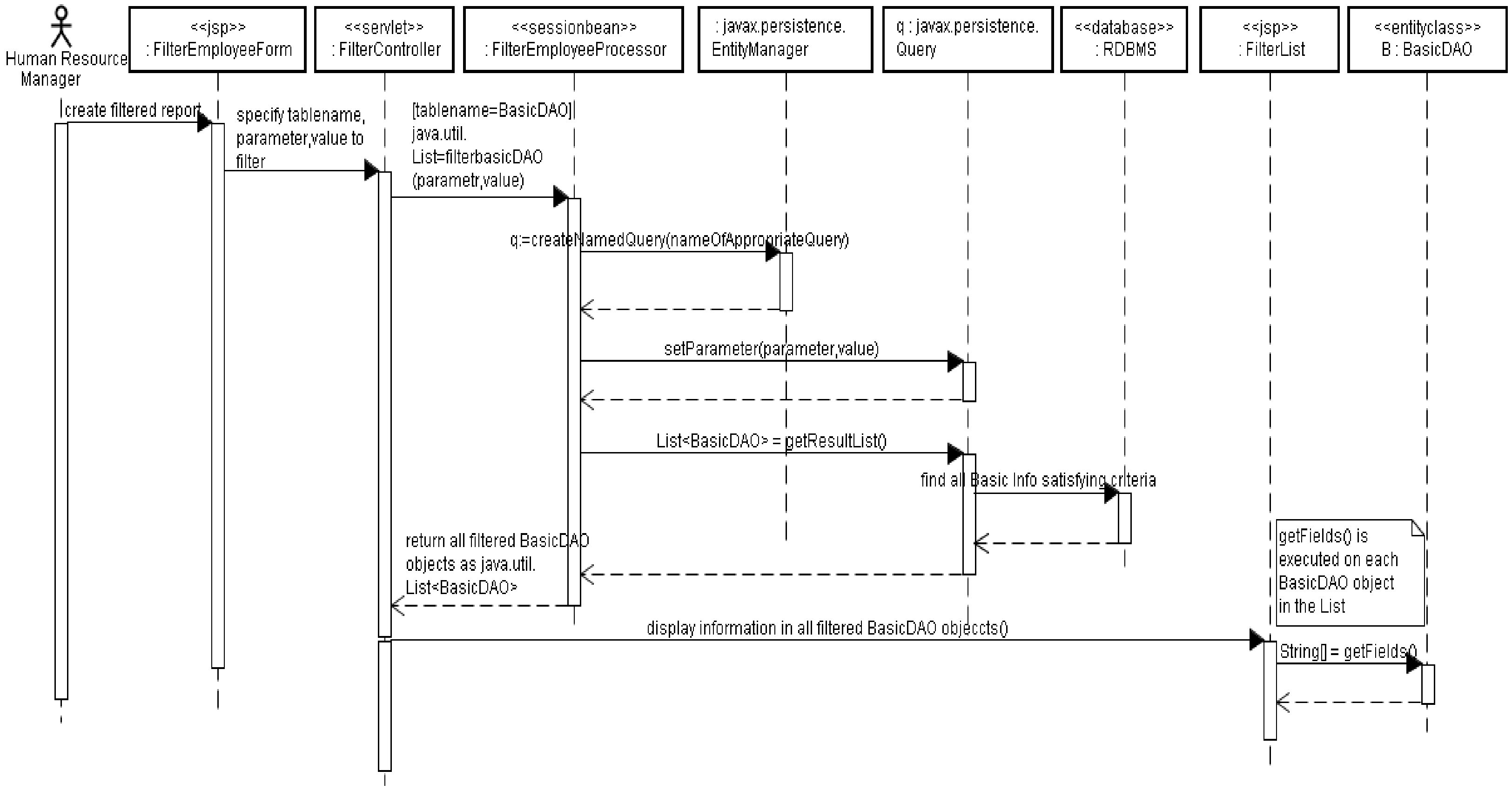

Human Resource Manager
<<jsp>> : FilterEmployeeForm
<<servlet>> : FilterController
<<sessionbean>> : FilterEmployeeProcessor
: javax.persistence. EntityManager
q : javax.persistence. Query
<<database>> : RDBMS
<<jsp>> : FilterList
<<entityclass>> B : BasicDAO
create filtered report
specify tablename, parameter,value to filter
[tablename=BasicDAO] java.util. List=filterbasicDAO (parametr,value)
q:=createNamedQuery(nameOfAppropriateQuery)
setParameter(parameter,value)
List<BasicDAO> = getResultList()
find all Basic Info satisfying criteria
return all filtered BasicDAO objects as java.util. List<BasicDAO>
getFields() is executed on each BasicDAO object in the List
display information in all filtered BasicDAO objeccts()
String[] = getFields()

Figure 30: create a filtered employee report using a table (BasicDAO table)

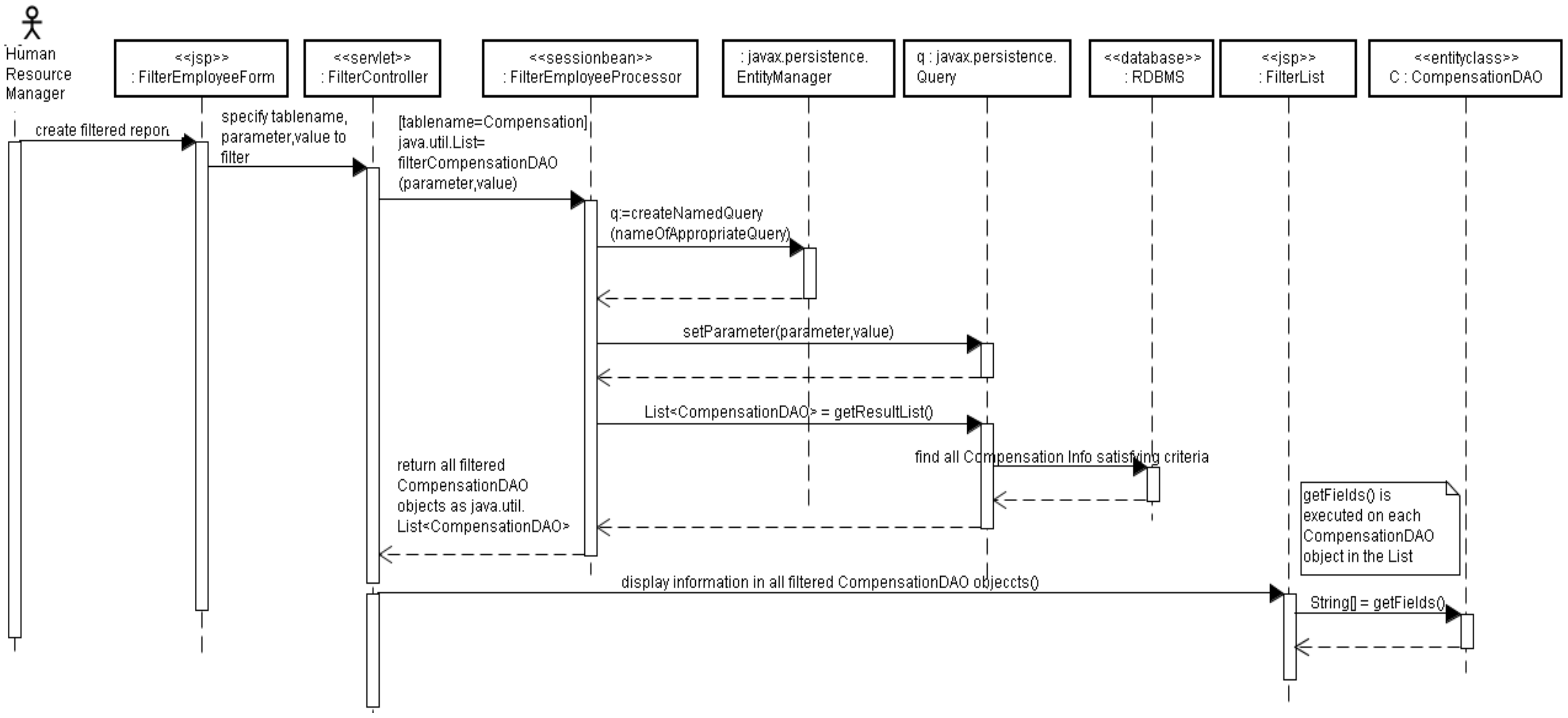

Human Resource Manager
<<jsp>> : FilterEmployeeForm
<<servlet>> : FilterController
<<sessionbean>> : FilterEmployeeProcessor
: javax.persistence. EntityManager
q : javax.persistence. Query
<<database>> : RDBMS
<<jsp>> : FilterList
<<entityclass>> C : CompensationDAO
create filtered report
specify tablename, parameter,value to filter
[tablename=Compensation] java.util.List= filterCompensationDAO (parameter,value)
q:=createNamedQuery (nameOfAppropriateQuery)
setParameter(parameter,value)
List<CompensationDAO> = getResultList()
find all Compensation Info satisfying criteria
return all filtered CompensationDAO objects as java.util. List<CompensationDAO>
getFields() is executed on each CompensationDAO object in the List
display information in all filtered CompensationDAO objeccts()
String[] = getFields()

Figure 31: create a filtered employee report using a table (CompensationDAO)

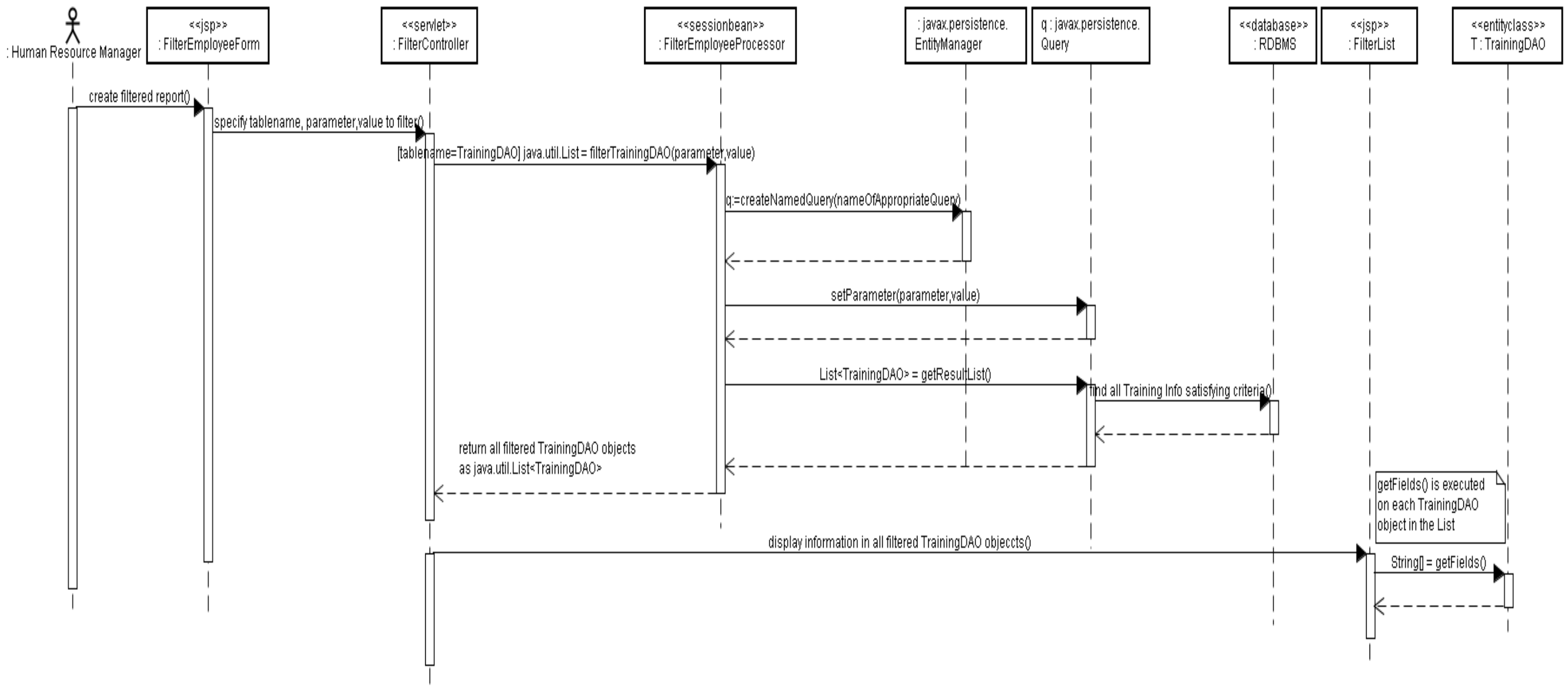

: Human Resource Manager
<<jsp>> : FilterEmployeeForm
<<servlet>> : FilterController
<<sessionbean>> : FilterEmployeeProcessor
: javax.persistence. EntityManager
q : javax.persistence. Query
<<database>> : RDBMS
<<jsp>> : FilterList
<<entityclass>> T : TrainingDAO
create filtered report()
specify tablename, parameter,value to filter()
[tablename=TrainingDAO] java.util.List = filterTrainingDAO(parameter,value)
q:=createNamedQuery(nameOfAppropriateQuery)
setParameter(parameter,value)
List<TrainingDAO> = getResultList()
find all Training Info satisfying criteria()
return all filtered TrainingDAO objects as java.util.List<TrainingDAO>
getFields() is executed on each TrainingDAO object in the List
display information in all filtered TrainingDAO objeccts()
String[] = getFields()

Figure 31: create a filtered employee report using a table (TrainingDAO) Please magnify & see. I did my best to make it more visible

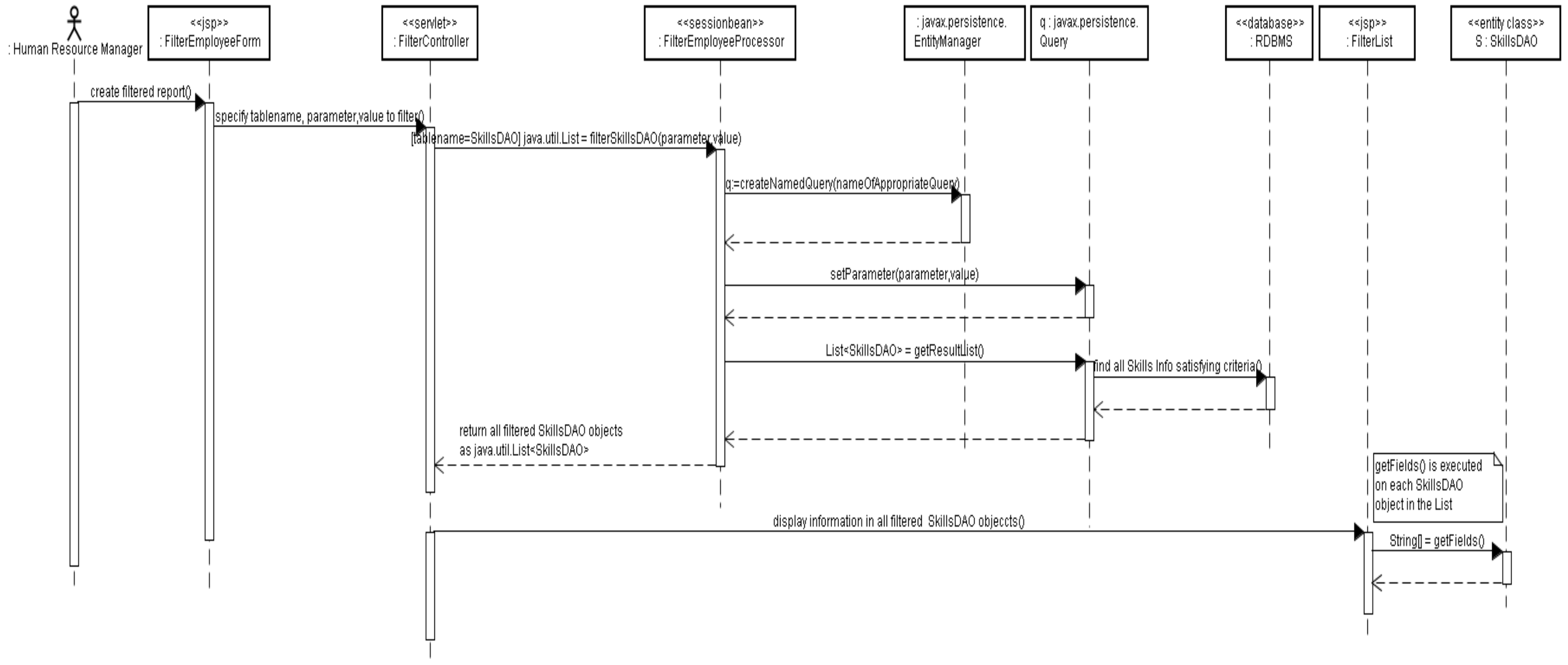

: Human Resource Manager
<<jsp>> : FilterEmployeeForm
<<servlet>> : FilterController
<<sessionbean>> : FilterEmployeeProcessor
: javax.persistence. EntityManager
q : javax.persistence. Query
<<database>> : RDBMS
<<jsp>> : FilterList
<<entity class>> S : SkillsDAO
create filtered report()
specify tablename, parameter,value to filter()
[tablename=SkillsDAO] java.util.List = filterSkillsDAO(parameter,value)
q:=createNamedQuery(nameOfAppropriateQuery)
setParameter(parameter,value)
List<SkillsDAO> = getResultList()
find all Skills Info satisfying criteria()
return all filtered SkillsDAO objects as java.util.List<SkillsDAO>
getFields() is executed on each SkillsDAO object in the List
display information in all filtered SkillsDAO objeccts()
String[] = getFields()

Figure 32: create a filtered employee report using a table (SkillsDAO) Please magnify.

-

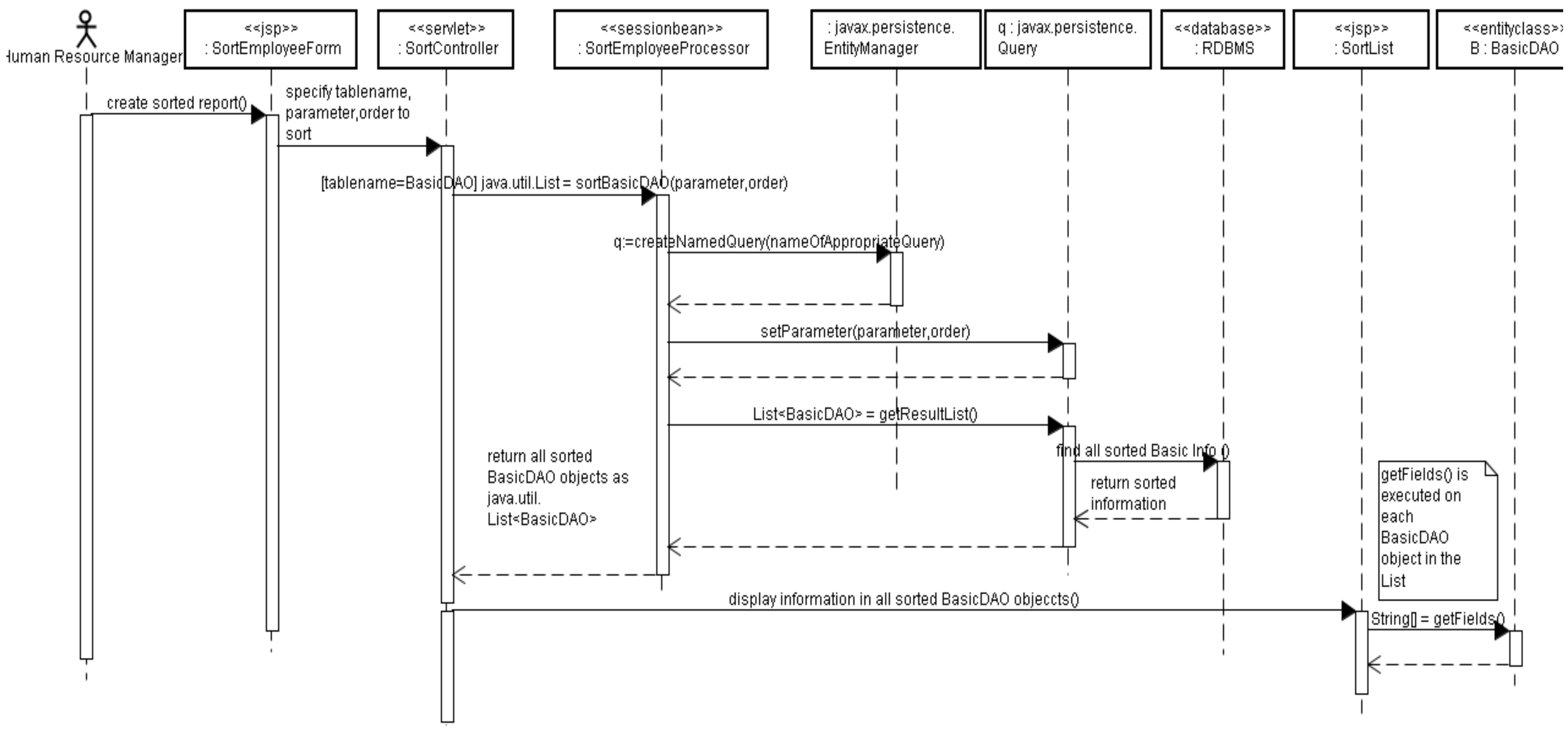

Human Resource Manager
<<jsp>> : SortEmployeeForm
<<servlet>> : SortController
<<sessionbean>> : SortEmployeeProcessor
: javax.persistence. EntityManager
q : javax.persistence. Query
<<database>> : RDBMS
<<jsp>> : SortList
<<entityclass>> B : BasicDAO
create sorted report()
specify tablename, parameter,order to sort
[tablename=BasicDAO] java.util.List = sortBasicDAO(parameter,order)
q:=createNamedQuery(nameOfAppropriateQuery)
setParameter(parameter,order)
List<BasicDAO> = getResultList()
find all sorted Basic Info ()
return sorted information
return all sorted BasicDAO objects as java.util. List<BasicDAO>
getFields() is executed on each BasicDAO object in the List
display information in all sorted BasicDAO objeccts()
String[] = getFields()

Figure 33: create a sorted employee report using a table (BasicDAO) Please magnify.

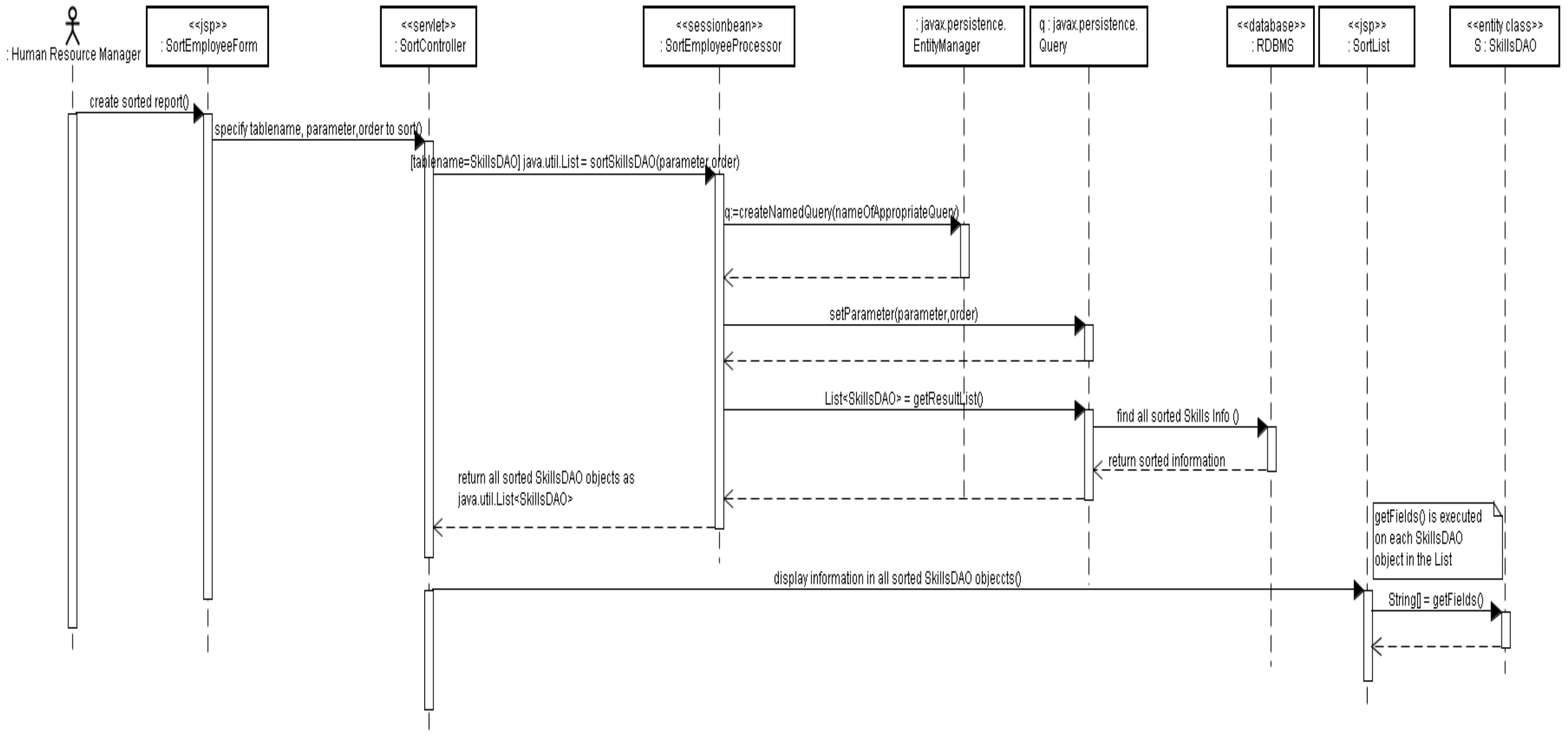

: Human Resource Manager
<<jsp>> : SortEmployeeForm
<<servlet>> : SortController
<<sessionbean>> : SortEmployeeProcessor
: javax.persistence. EntityManager
q : javax.persistence. Query
<<database>> : RDBMS
<<jsp>> : SortList
<<entity class>> S : SkillsDAO
create sorted report()
specify tablename, parameter,order to sort()
[tablename=SkillsDAO] java.util.List = sortSkillsDAO(parameter,order)
q:=createNamedQuery(nameOfAppropriateQuery)
setParameter(parameter,order)
List<SkillsDAO> = getResultList()
find all sorted Skills Info ()
return sorted information
return all sorted SkillsDAO objects as java.util.List<SkillsDAO>
getFields() is executed on each SkillsDAO object in the List
display information in all sorted SkillsDAO objeccts()
String[] = getFields()

Figure 33: create a sorted employee report using a table (SkillsDAO) Please magnify.

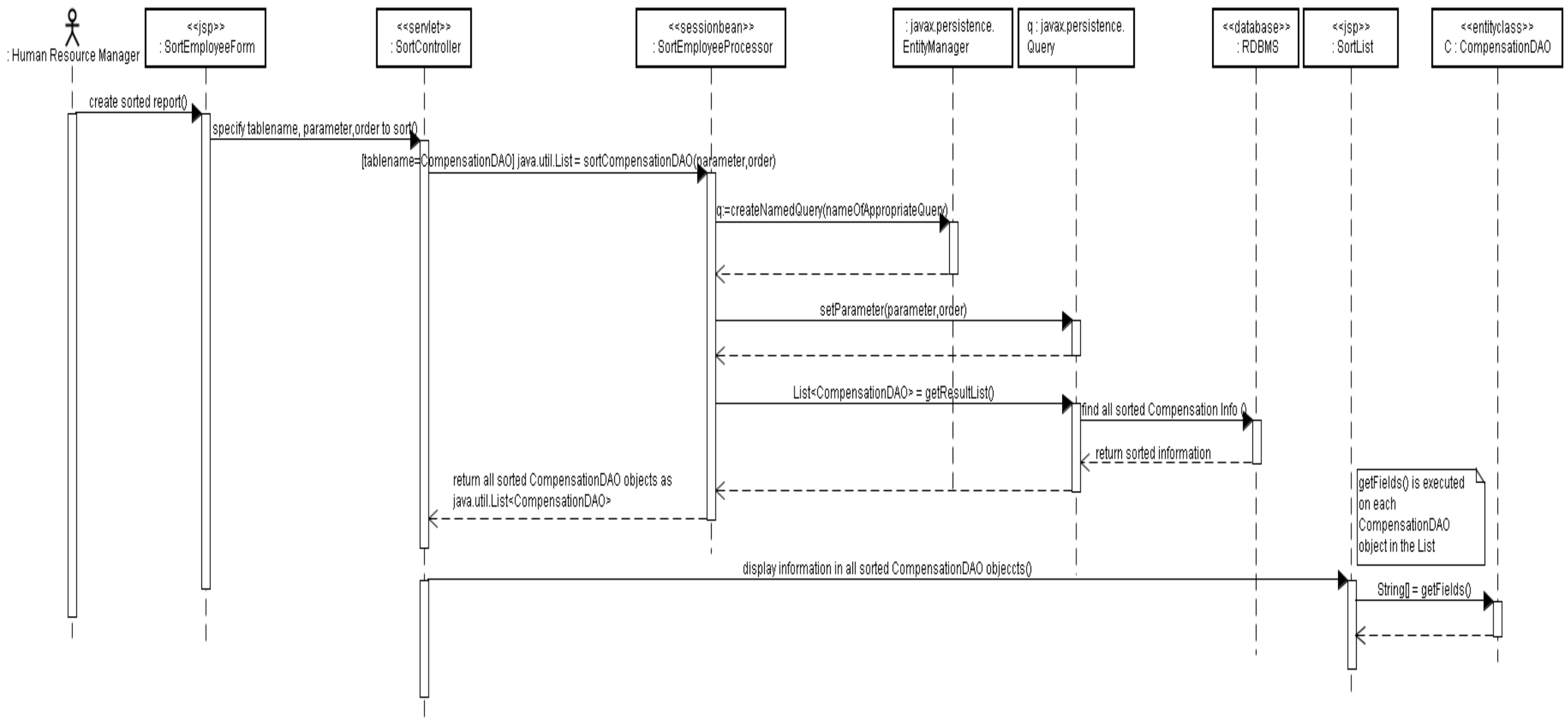

: Human Resource Manager
<<jsp>> : SortEmployeeForm
<<servlet>> : SortController
<<sessionbean>> : SortEmployeeProcessor
: javax.persistence. EntityManager
q : javax.persistence. Query
<<database>> : RDBMS
<<jsp>> : SortList
<<entityclass>> C : CompensationDAO
create sorted report()
specify tablename, parameter,order to sort()
[tablename=CompensationDAO] java.util.List = sortCompensationDAO(parameter,order)
q:=createNamedQuery(nameOfAppropriateQuery)
setParameter(parameter,order)
List<CompensationDAO> = getResultList()
find all sorted Compensation Info ()
return sorted information
return all sorted CompensationDAO objects as java.util.List<CompensationDAO>
getFields() is executed on each CompensationDAO object in the List
display information in all sorted CompensationDAO objeccts()
String[] = getFields()

Figure 34: create a sorted employee report using a table (CompensationDAO) Please magnify.

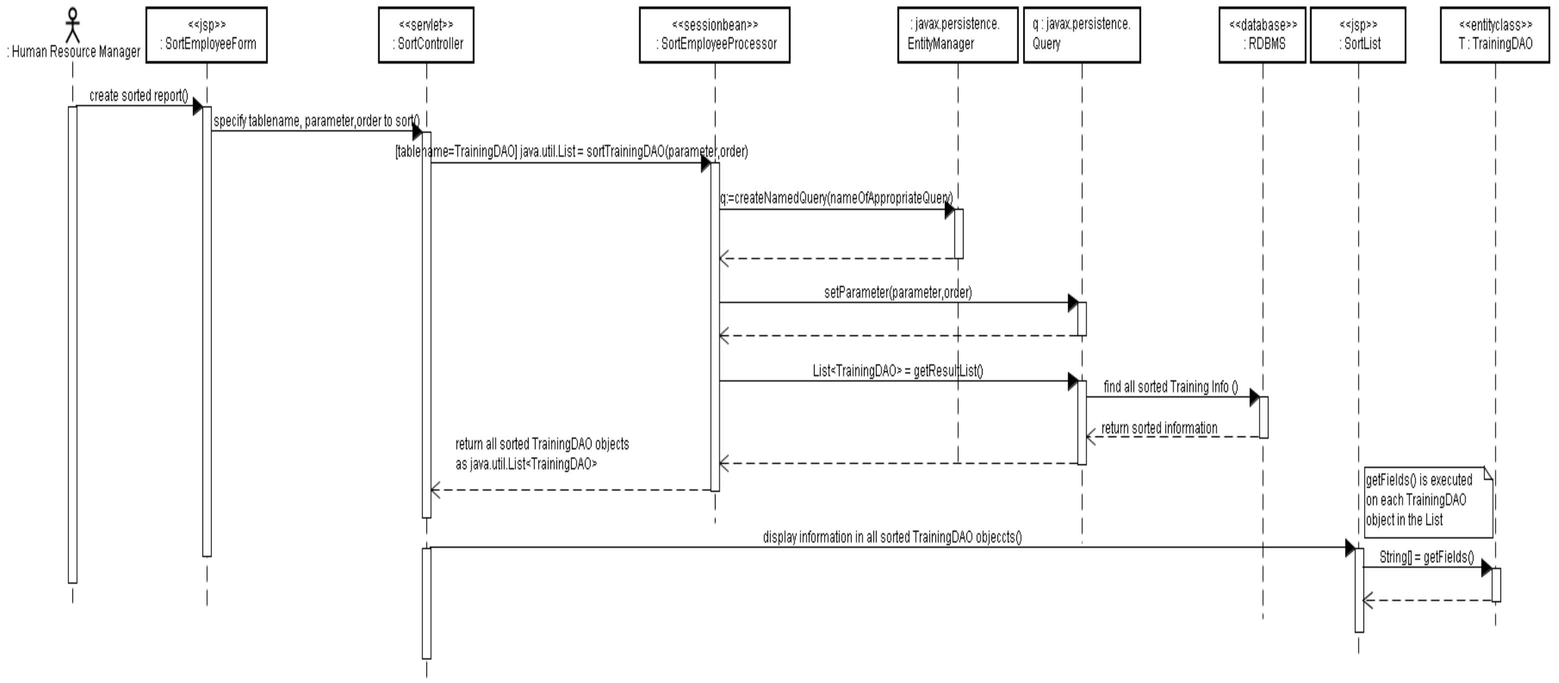

: Human Resource Manager
<<jsp>> : SortEmployeeForm
<<servlet>> : SortController
<<sessionbean>> : SortEmployeeProcessor
: javax.persistence. EntityManager
q : javax.persistence. Query
<<database>> : RDBMS
<<jsp>> : SortList
<<entityclass>> T : TrainingDAO
create sorted report()
specify tablename, parameter,order to sort()
[tablename=TrainingDAO] java.util.List = sortTrainingDAO(parameter,order)
q:=createNamedQuery(nameOfAppropriateQuery)
setParameter(parameter,order)
List<TrainingDAO> = getResultList()
find all sorted Training Info ()
return sorted information
return all sorted TrainingDAO objects as java.util.List<TrainingDAO>
getFields() is executed on each TrainingDAO object in the List
display information in all sorted TrainingDAO objeccts()
String[] = getFields()

Figure 34: create a sorted employee report using a table (TrainingDAO) Please magnify.

-

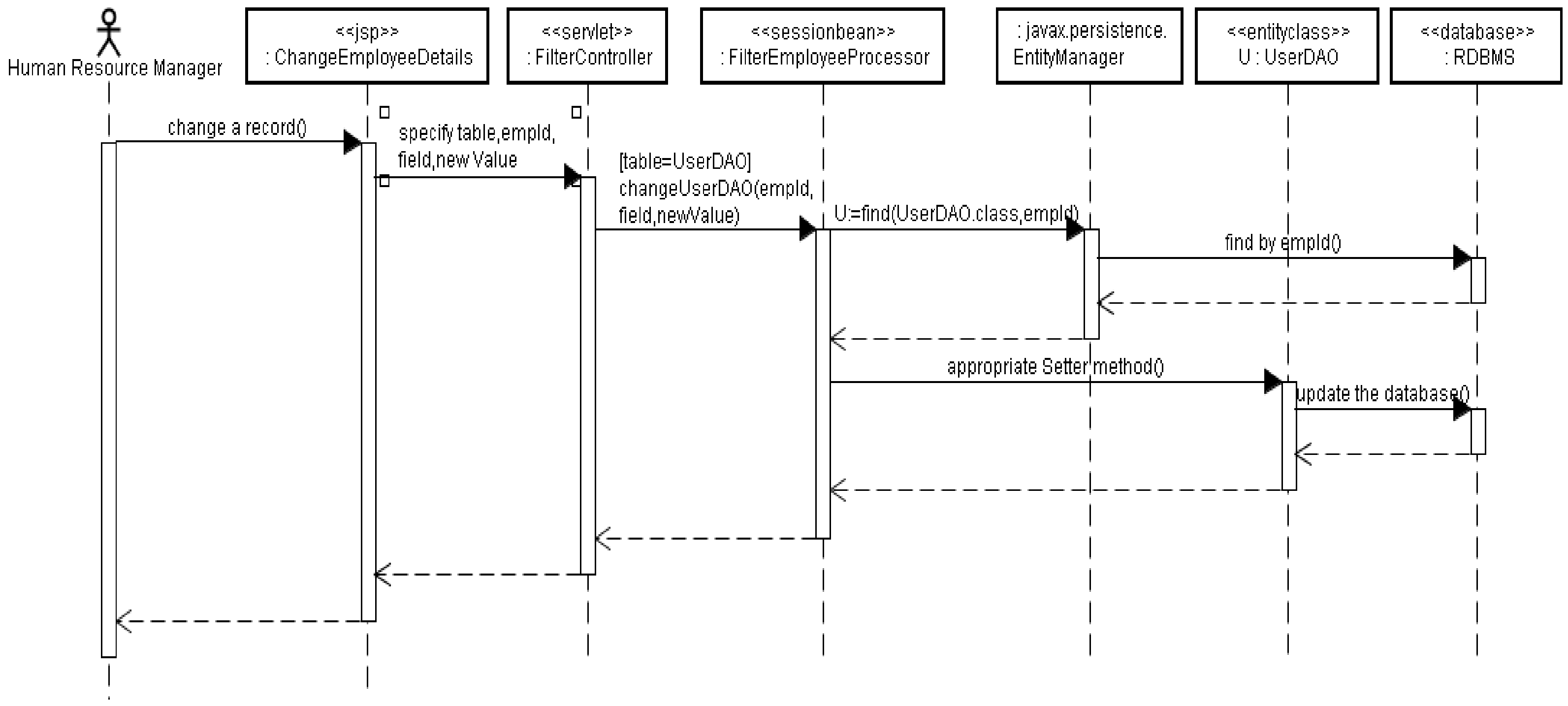

Human Resource Manager
<<jsp>> : ChangeEmployeeDetails
<<servlet>> : FilterController
<<sessionbean>> : FilterEmployeeProcessor
: javax.persistence. EntityManager
<<entityclass>> U : UserDAO
<<database>> : RDBMS
change a record()
specify table,empId, field,new Value
[table=UserDAO] changeUserDAO(empId, field,newValue)
U:=find(UserDAO.class,empId)
find by empId()
appropriate Setter method()
update the database()

Figure 34: change employee records from table BasicDAO

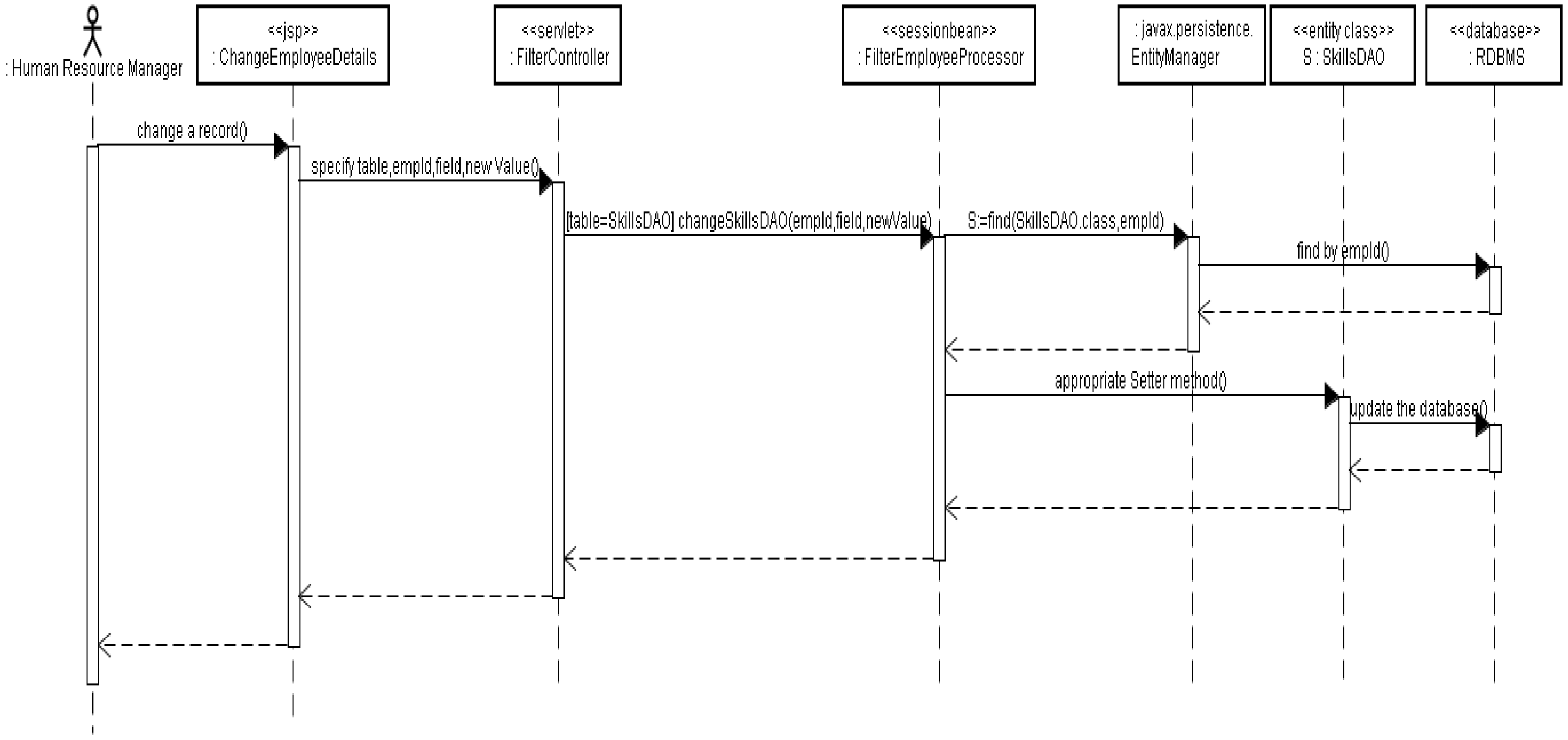

: Human Resource Manager
<<jsp>> : ChangeEmployeeDetails
<<servlet>> : FilterController
<<sessionbean>> : FilterEmployeeProcessor
: javax.persistence. EntityManager
<<entity class>> S : SkillsDAO
<<database>> : RDBMS
change a record()
specify table,empId,field,new Value()
[table=SkillsDAO] changeSkillsDAO(empId,field,newValue)
S:=find(SkillsDAO.class,empId)
find by empId()
appropriate Setter method()
update the database()

Figure 34: change employee records from table SkillsDAO(Please magnify)

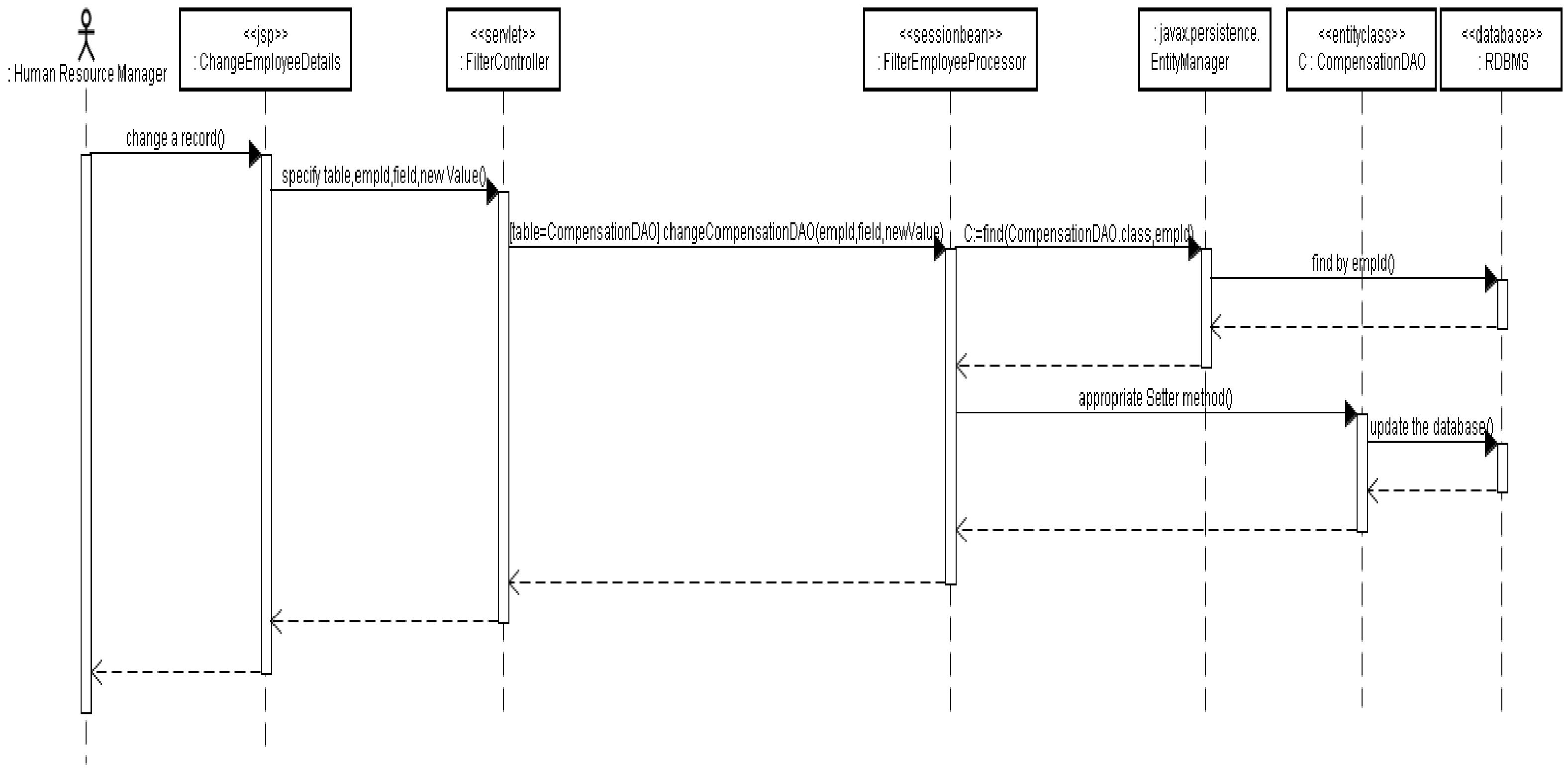

: Human Resource Manager
<<jsp>> : ChangeEmployeeDetails
<<servlet>> : FilterController
<<sessionbean>> : FilterEmployeeProcessor
: javax.persistence. EntityManager
<<entityclass>> C : CompensationDAO
<<database>> : RDBMS
change a record()
specify table,empId,field,new Value()
[table=CompensationDAO] changeCompensationDAO(empId,field,newValue)
C:=find(CompensationDAO.class,empId)
find by empId()
appropriate Setter method()
update the database()

Figure 34: change employee records from table CompensationDAO (Please magnify)

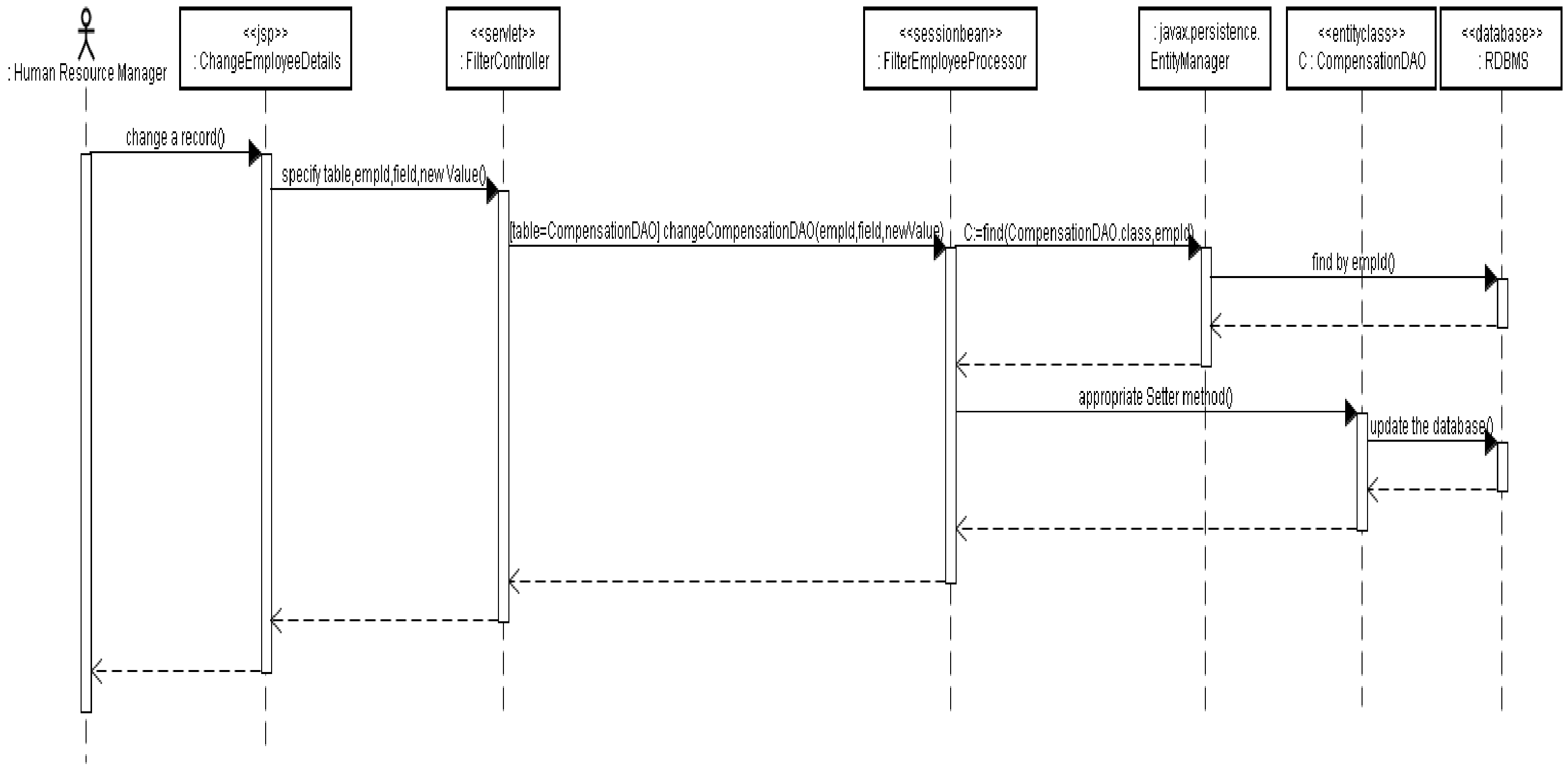

: Human Resource Manager
<<jsp>> : ChangeEmployeeDetails
<<servlet>> : FilterController
<<sessionbean>> : FilterEmployeeProcessor
: javax.persistence. EntityManager
<<entityclass>> C : CompensationDAO
<<database>> : RDBMS
change a record()
specify table,empId,field,new Value()
[table=CompensationDAO] changeCompensationDAO(empId,field,newValue)
C:=find(CompensationDAO.class,empId)
find by empId()
appropriate Setter method()
update the database()

Figure 35: change employee records from table TrainingDAO (Please magnify)

-

3.2.3 Architectural modeling

I modeled how the developed logical constituents are going to be implemented in the physical system. Further more I wanted to model in which physical destination each component resides. So I drew following diagrams

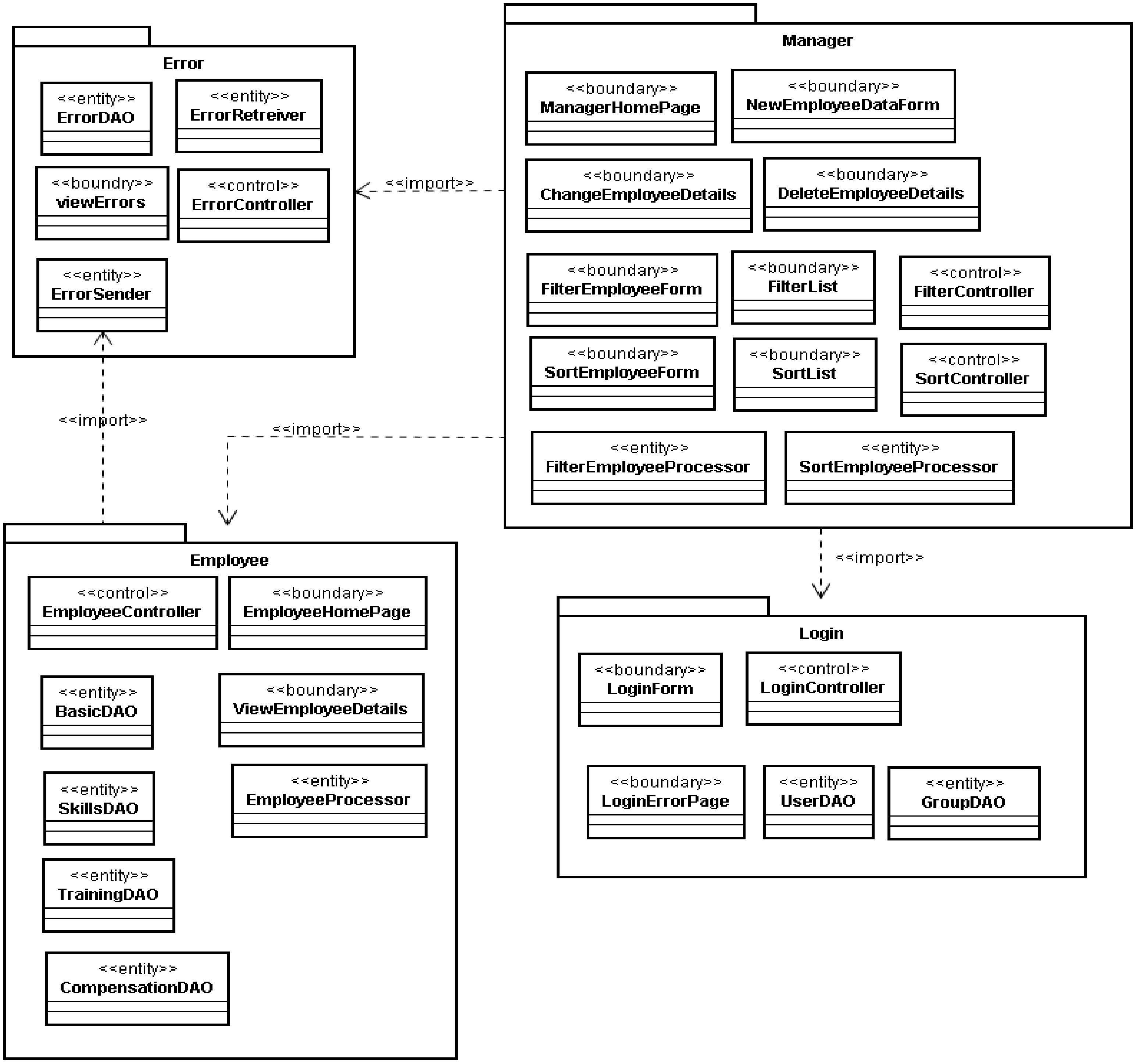

Error
<<entity>> ErrorDAO
<<entity>> ErrorRetreiver
<<boundry>> viewErrors
<<control>> ErrorController
<<entity>> ErrorSender
<<import>>
<<import>>
<<import>>
<<import>>
Manager
<<boundary>> ManagerHomePage
<<boundary>> NewEmployeeDataForm
<<boundary>> ChangeEmployeeDetails
<<boundary>> DeleteEmployeeDetails
<<boundary>> FilterEmployeeForm
<<boundary>> FilterList
<<control>> FilterController
<<boundary>> SortEmployeeForm
<<boundary>> SortList
<<control>> SortController
<<entity>> FilterEmployeeProcessor
<<entity>> SortEmployeeProcessor
Employee
<<control>> EmployeeController
<<boundary>> EmployeeHomePage
<<entity>> BasicDAO
<<boundary>> ViewEmployeeDetails
<<entity>> SkillsDAO
<<entity>> EmployeeProcessor
<<entity>> TrainingDAO
<<entity>> CompensationDAO
Login
<<boundary>> LoginForm
<<control>> LoginController
<<boundary>> LoginErrorPage
<<entity>> UserDAO
<<entity>> GroupDAO

Figure 35: package diagram for proposed solution

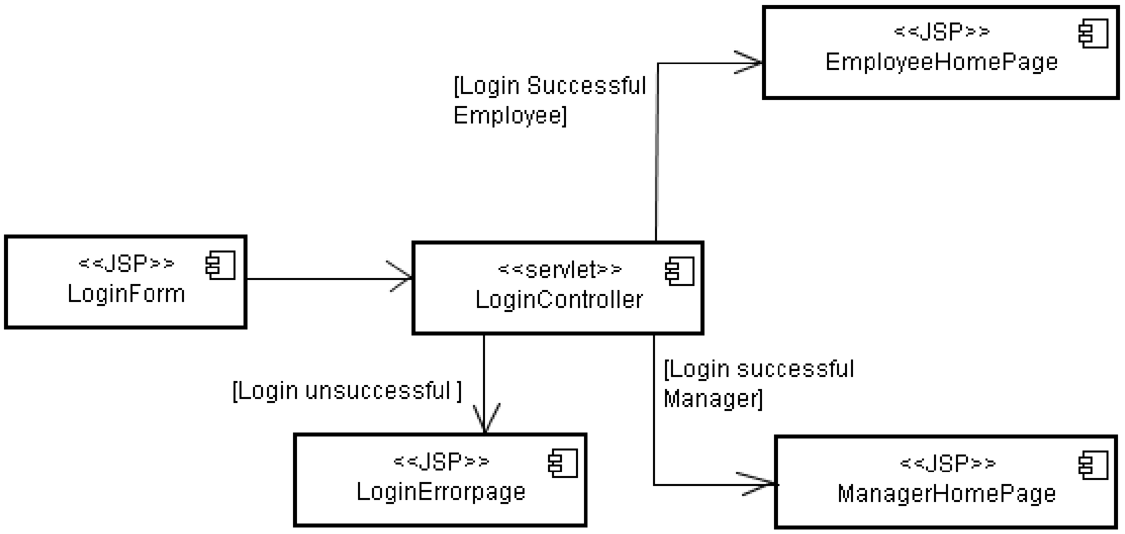


Figure 36: login component diagram

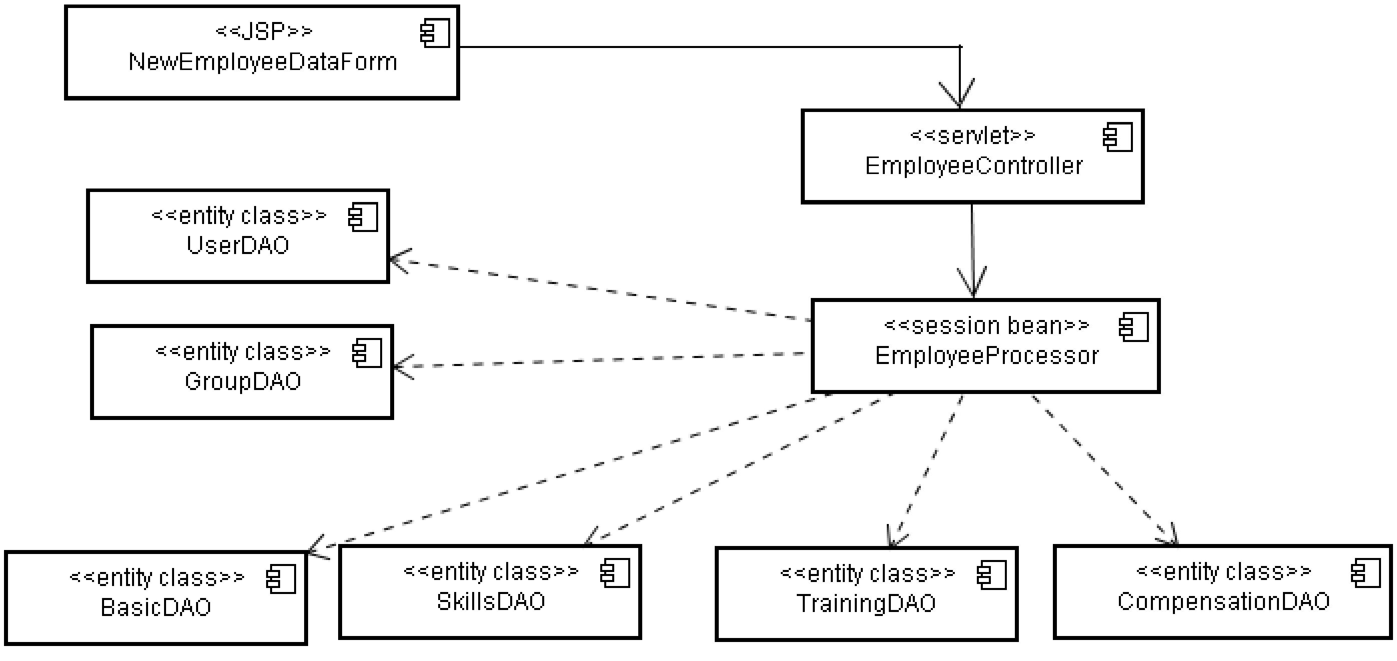


Figure 36: store a new employee record component diagram

-

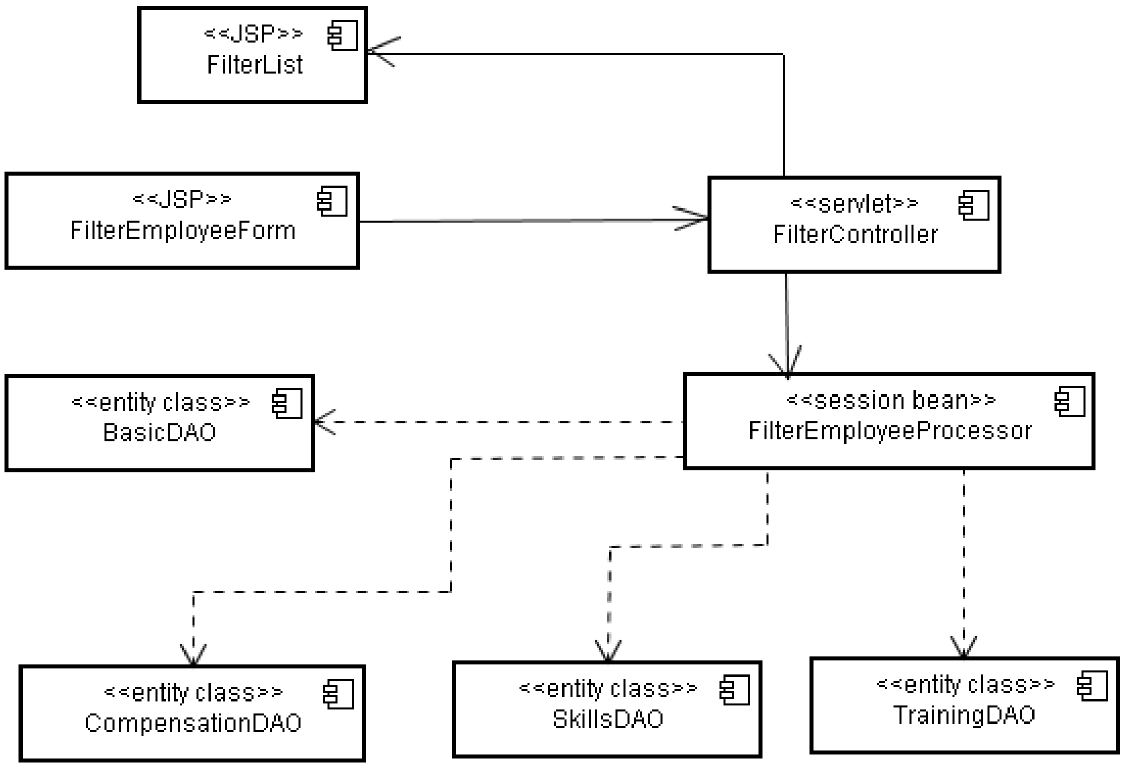


Figure 37: create a filtered employee report component diagram

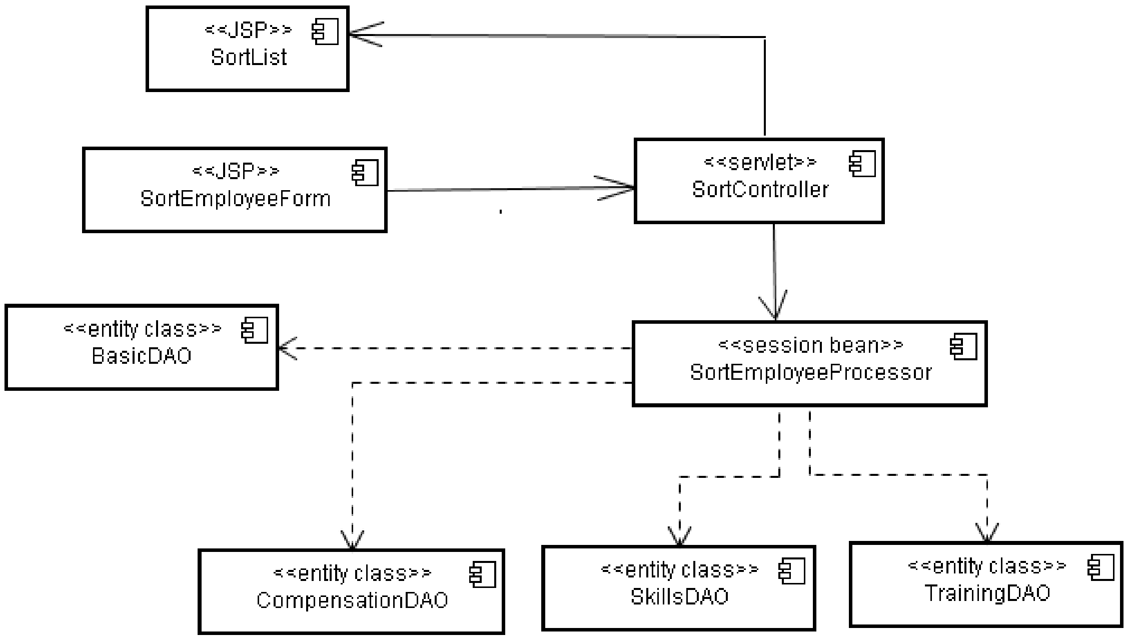


Figure 38: create a sorted employee report component diagram

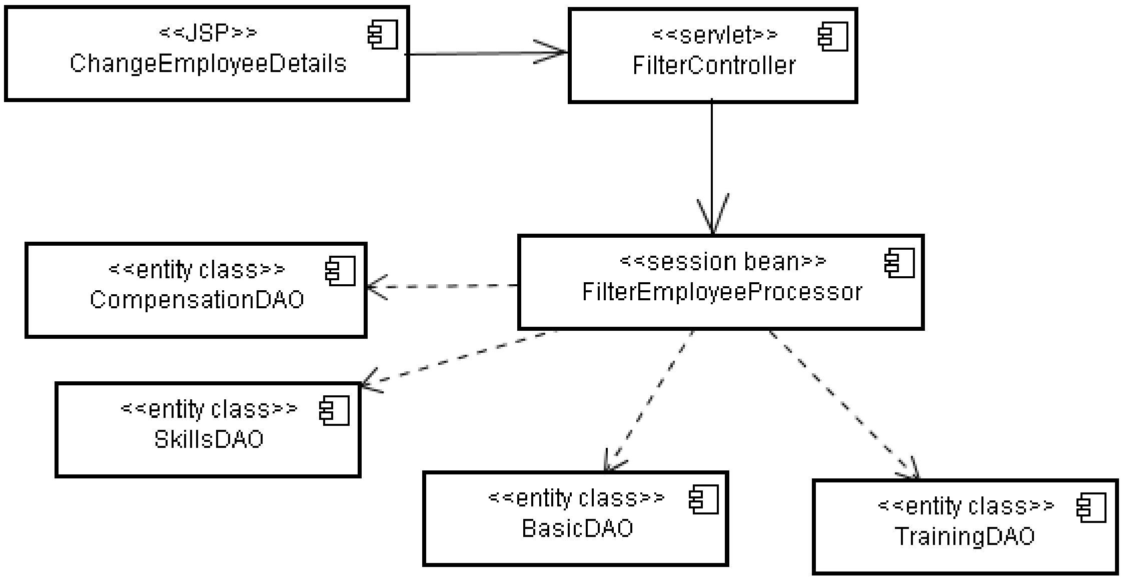


Figure 39: change employee details component diagram

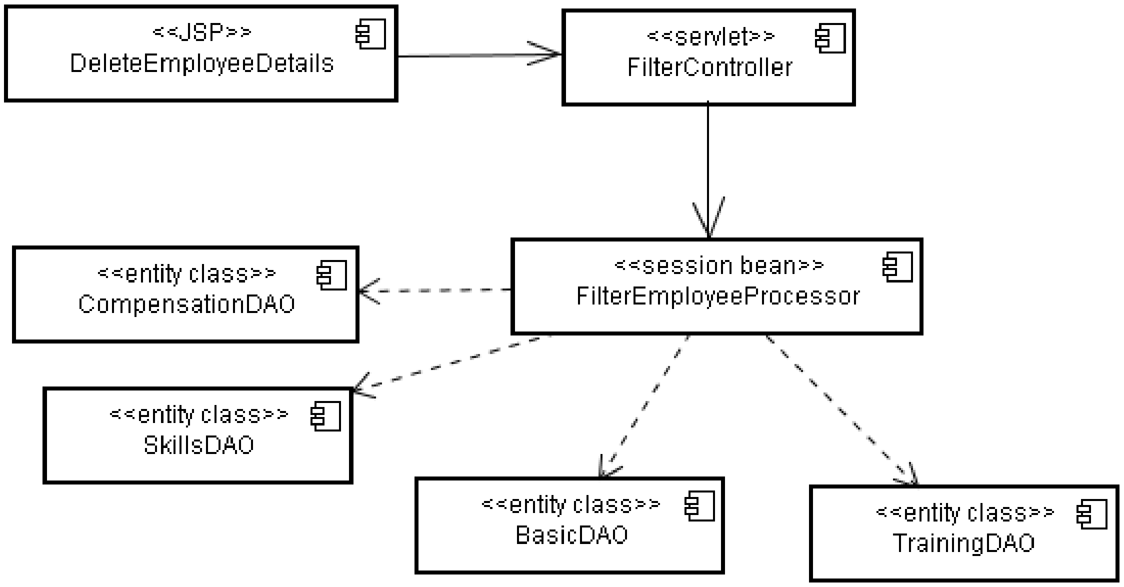


Figure 40: delete employee details component diagram

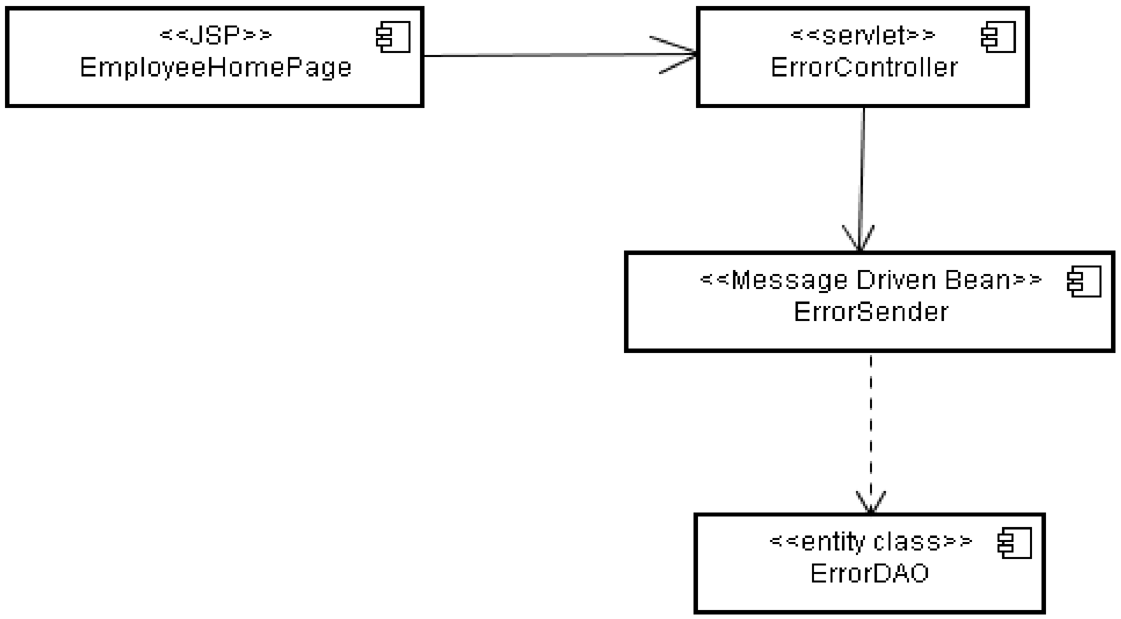


Figure 40: report errors component diagram

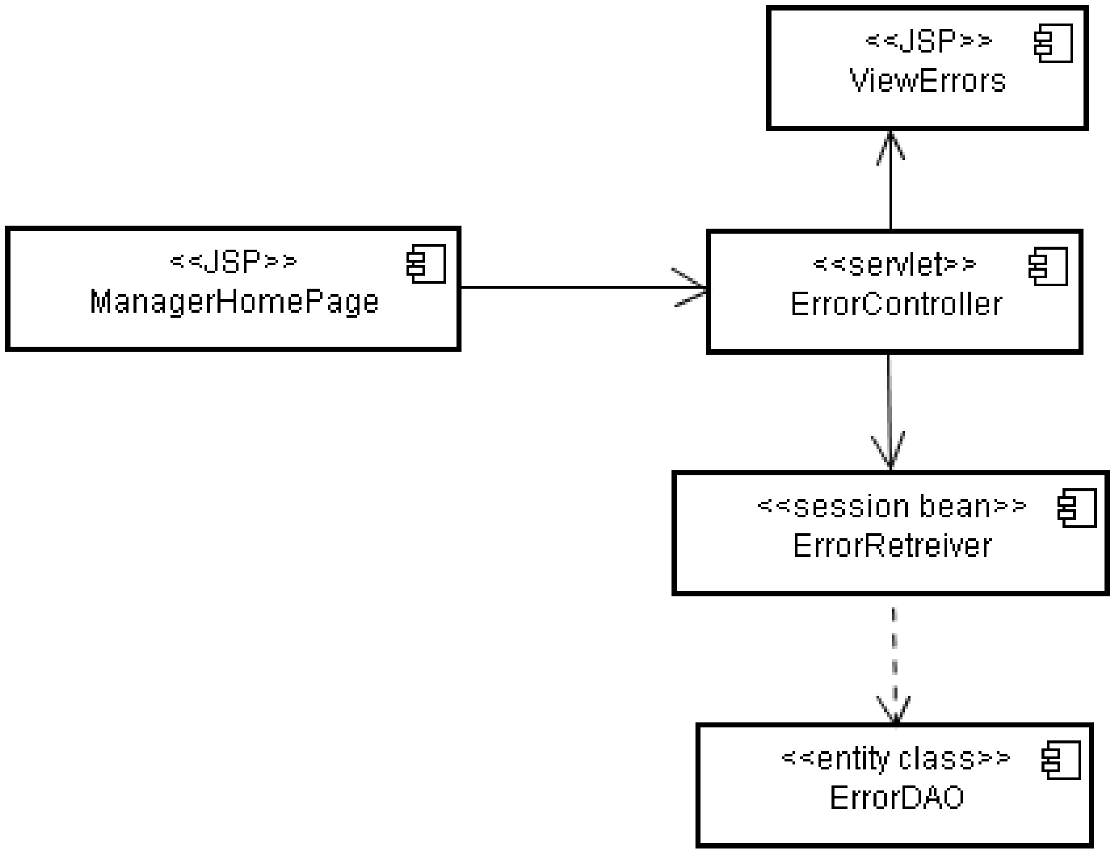


Figure 41: view errors reported by employees component diagram

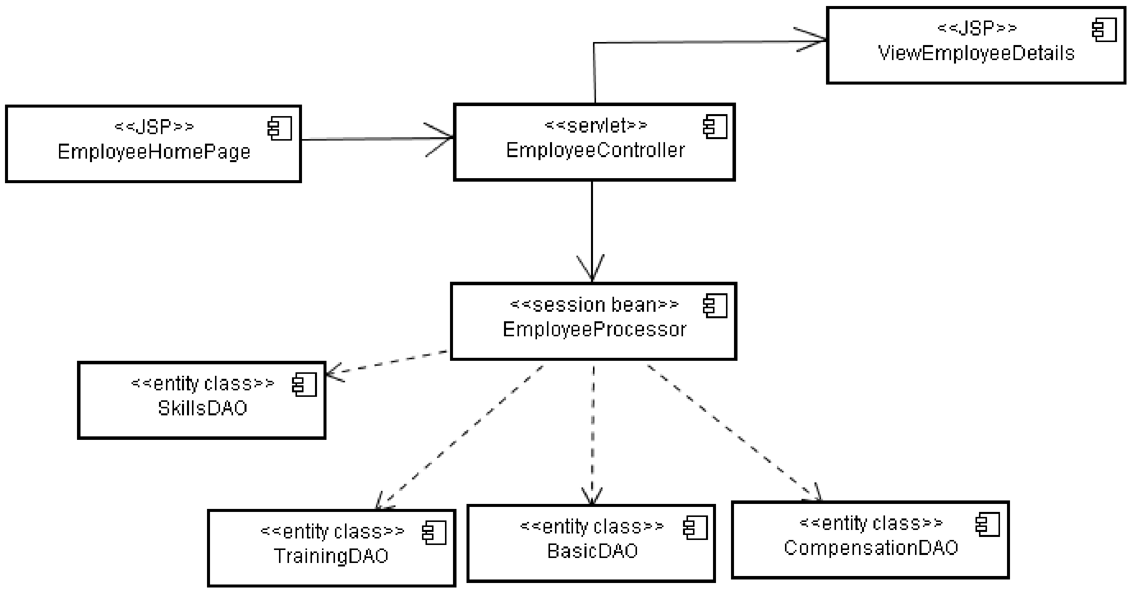


Figure 41: view employee information component diagram

-

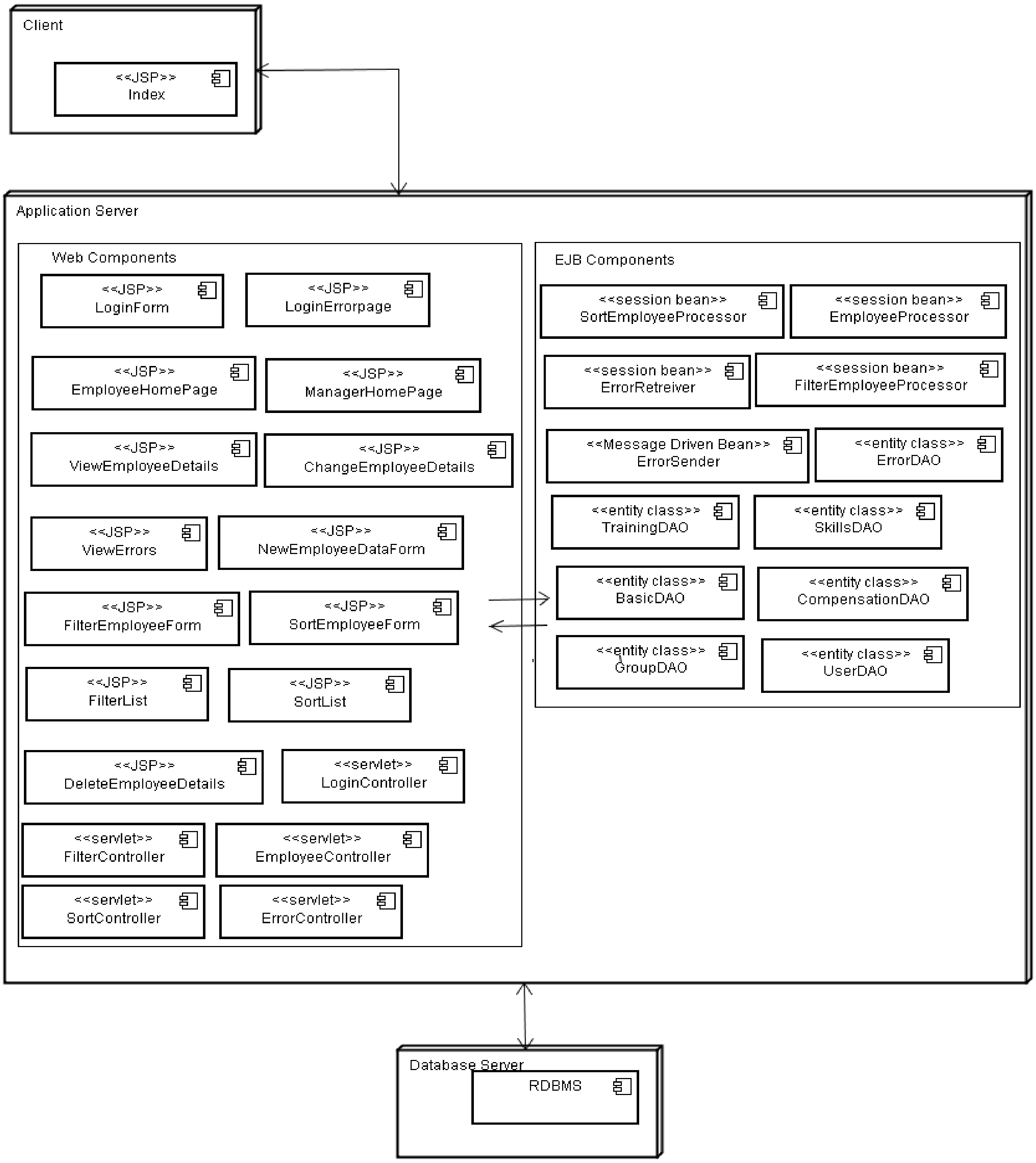

Client
<<JSP>> Index
Application Server
Web Components
<<JSP>> LoginForm
<<JSP>> LoginErrorpage
<<JSP>> EmployeeHomePage
<<JSP>> ManagerHomePage
<<JSP>> ViewEmployeeDetails
<<JSP>> ChangeEmployeeDetails
<<JSP>> ViewErrors
<<JSP>> NewEmployeeDataForm
<<JSP>> FilterEmployeeForm
<<JSP>> SortEmployeeForm
<<JSP>> FilterList
<<JSP>> SortList
<<JSP>> DeleteEmployeeDetails
<<servlet>> LoginController
<<servlet>> FilterController
<<servlet>> EmployeeController
<<servlet>> SortController
<<servlet>> ErrorController
EJB Components
<<session bean>> SortEmployeeProcessor
<<session bean>> EmployeeProcessor
<<session bean>> ErrorRetreiver
<<session bean>> FilterEmployeeProcessor
<<Message Driven Bean>> ErrorSender
<<entity class>> ErrorDAO
<<entity class>> TrainingDAO
<<entity class>> SkillsDAO
<<entity class>> BasicDAO
<<entity class>> CompensationDAO
<<entity class>> GroupDAO
<<entity class>> UserDAO
Database Server
RDBMS

Figure 42: deployment diagram

3.2.4 Coding Phase

- Entity classes

  I developed an entity class that matches with each & every table as I said earlier. I declared named Queries for each & every field in the table. In all entity classes I declared a method to retrieve the values of all member variables at once. I am putting the coding for one of the entity classes I developed in the ***Appendix 5***.

- Session beans

  For each session bean there is a business interface & implementing class. As I mentioned under section 2 Literature I had methods to manage the entities from the session bean class. I have described how I did earlier. I am putting coding for one of the developed session bean class as well as its business interface in the ***Appendix 6***.

- Message driven bean.

  I had only one message driven bean & A Servlet class was sending messages to the message driven bean. I have discussed how I did in detail earlier. I am putting coding for the message driven bean & the servlet class sending messages to the message driven bean in the ***Appendix 7***.

- Servlet classes

  In the `Login controller` servlet class forwarding the users to their corresponding home pages is done. I am mentioning that class in ***Appendix 8*** since I couldn't explain about it more. One of another normal servlet classes also I will add to ***appendix 9***.

- Java Server Pages
  Coding for one of the developed Java Server page also I will add to ***Appendix 10***.

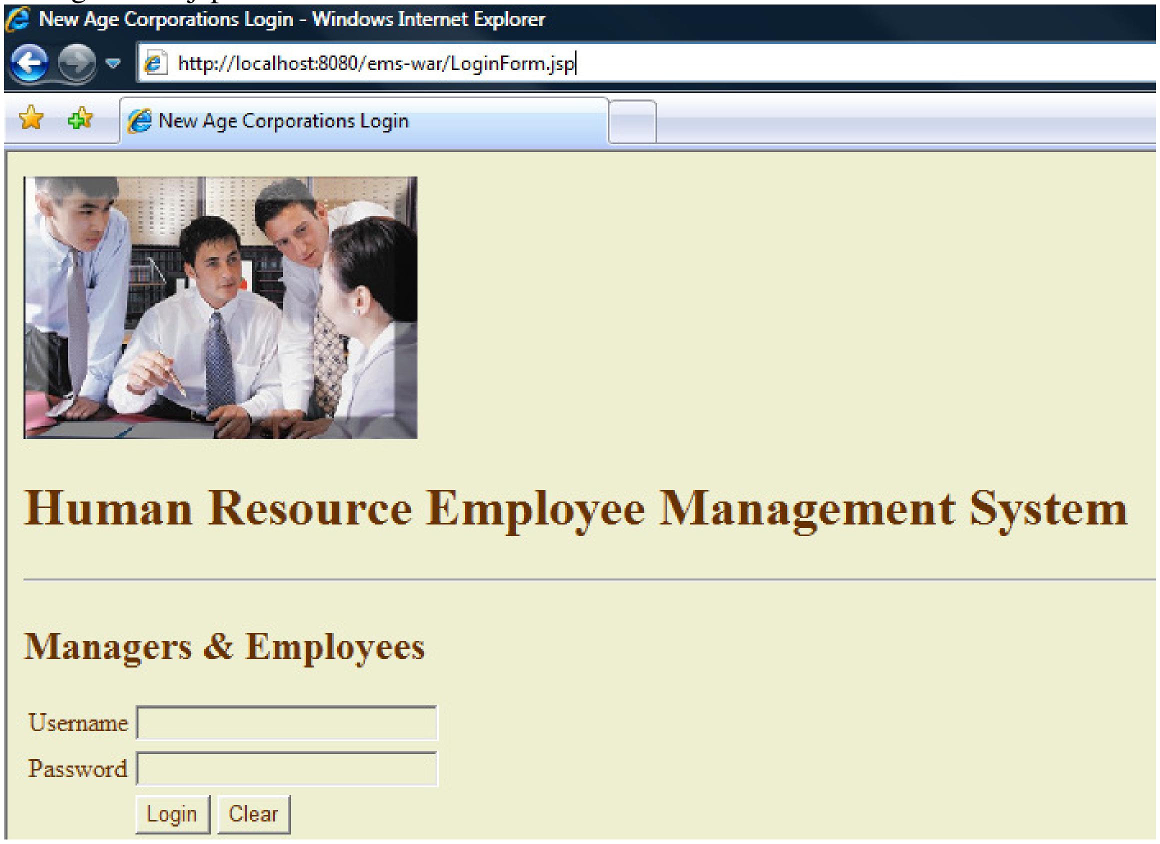
-

3.2.5 Modeling graphical user interfaces

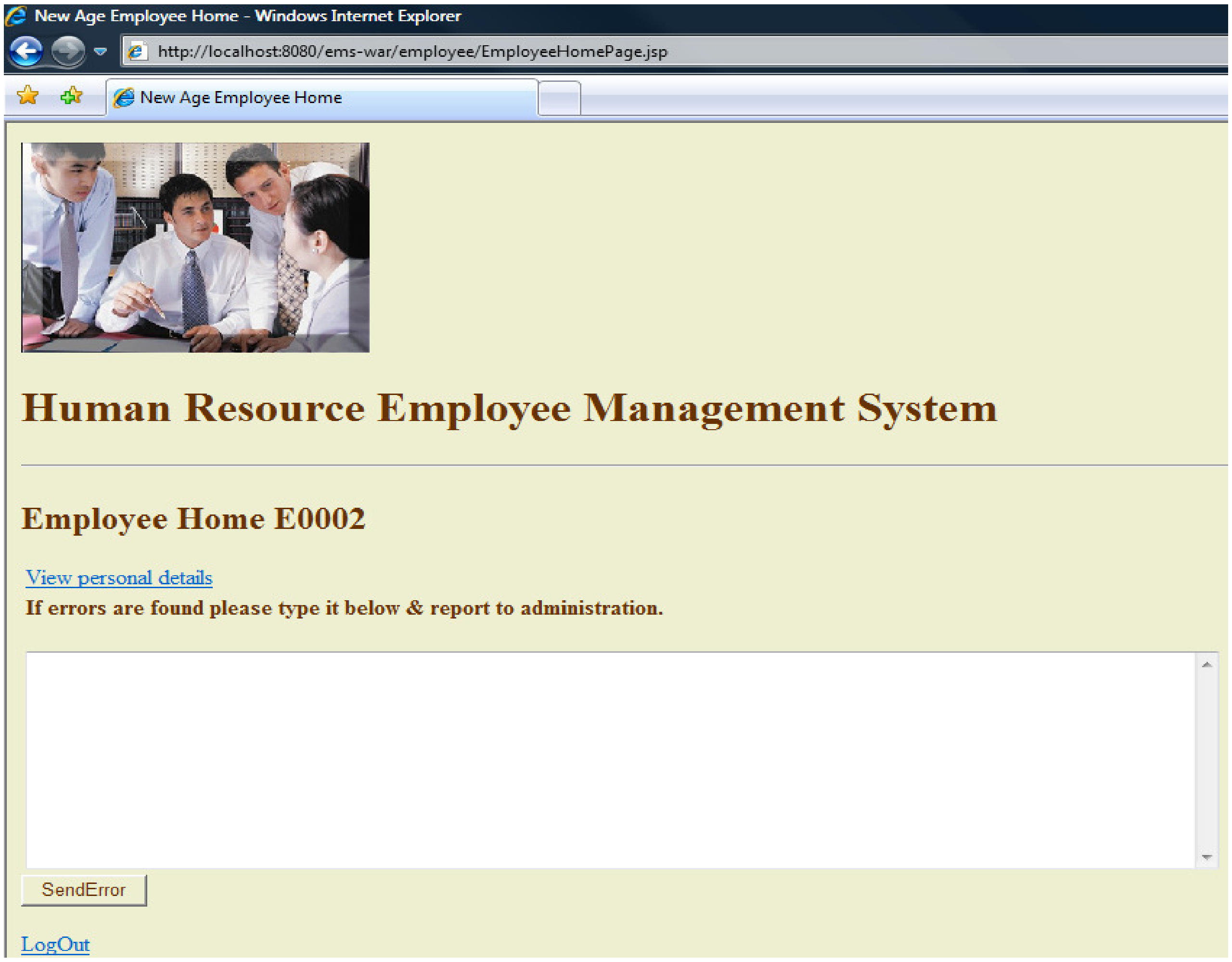
I did modeling the graphical user interfaces with the help of Macromedia Dream Viewer. In front of the human resource managers I developed the user interfaces. In Macromedia developing a web page means most of the time just dragging & dropping. So after developing I show it to the human resource managers. Then they suggested the changes they wish to see. So by the way at last a final version of the web page derived.
Some of the previously created user interfaces but changed according to human resource manager's feedback are included in the ***Appendix 11***.
One by one I am giving below screen shots of the final versions of the Java Server Pages built.

<u>Employee's role</u>

1. LoginForm.jsp


New Age Corporations Login - Windows Internet Explorer
http://localhost:8080/ems-war/LoginForm.jsp
New Age Corporations Login
Human Resource Employee Management System
Managers & Employees
Username
Password
Login
Clear

2.If the user is an employee after login he is forwarded to employee home page. (`EmployeeHomePage.jsp`).


New Age Employee Home - Windows Internet Explorer
http://localhost:8080/ems-war/employee/EmployeeHomePage.jsp
New Age Employee Home
SendError

Human Resource Employee Management System

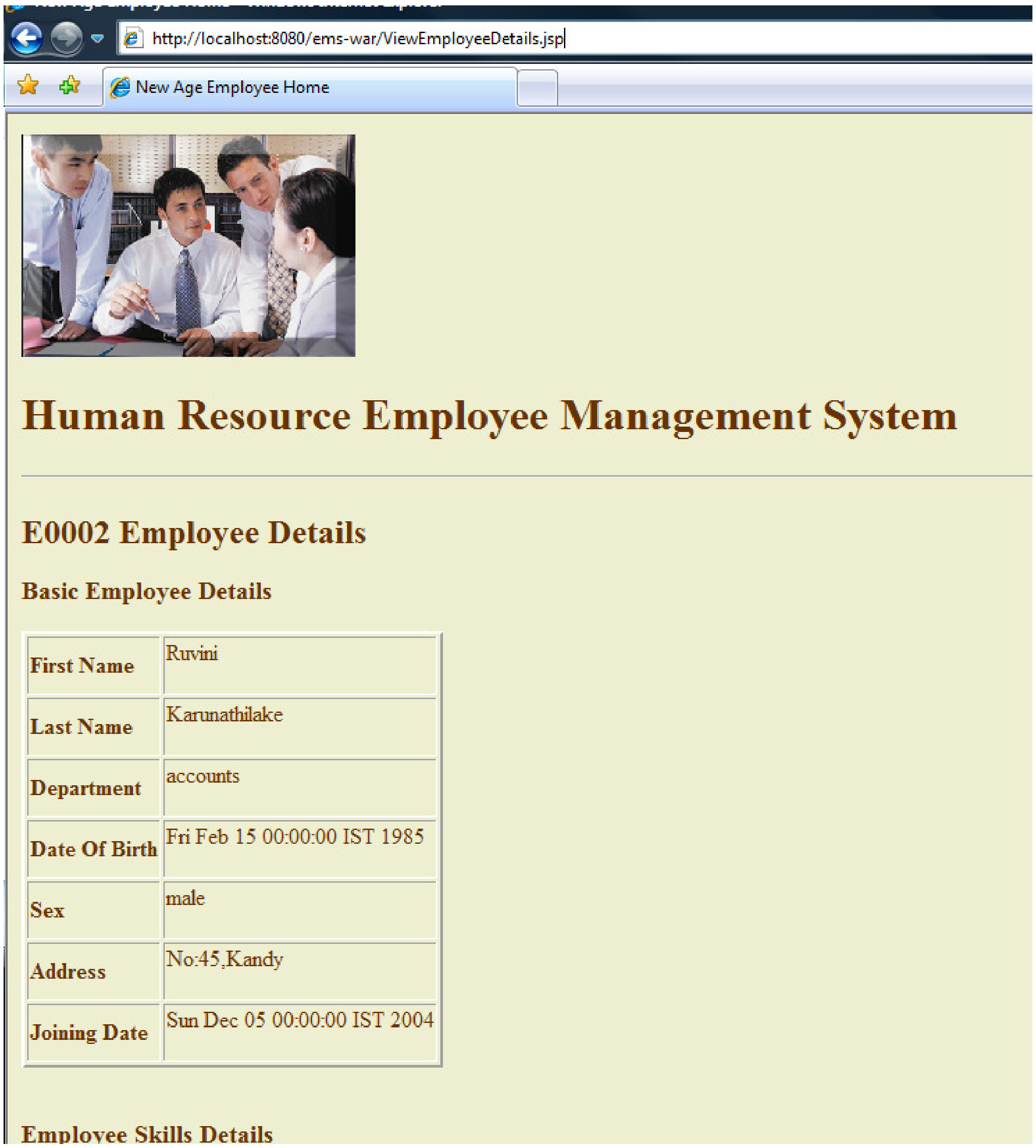
Employee Home E0002

View personal details

If errors are found please type it below & report to administration.

LogOut

3.Then if he/she wants to know about his/her employee information he/she can click on view employee details link. It's will display the employee information in `ViewEmployeeDetails.jsp` JSP page.



Human Resource Employee Management System

E0002 Employee Details

Basic Employee Details

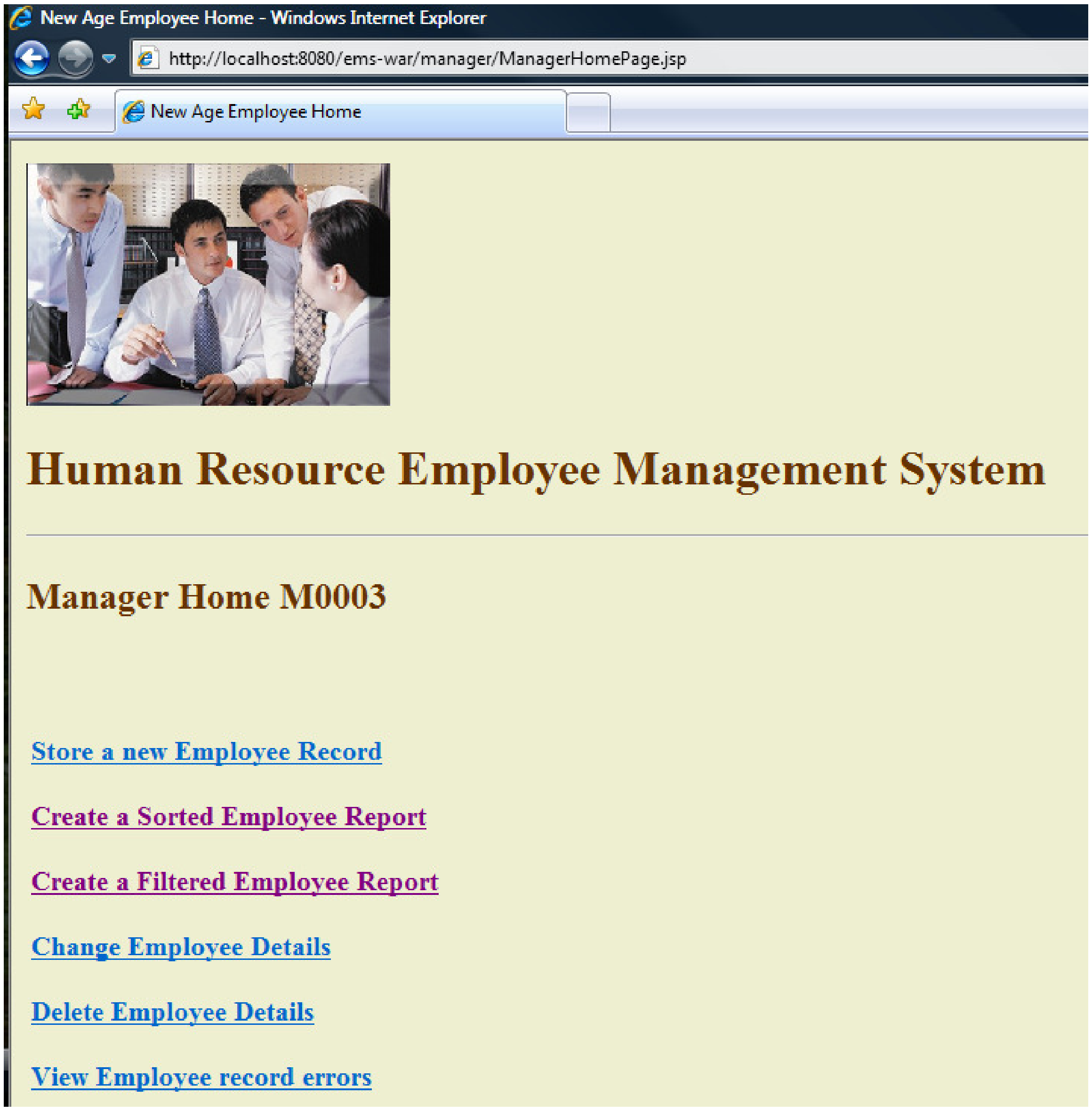
First Name	Ruvini
Last Name	Karunathilake
Department	accounts
Date Of Birth	Fri Feb 15 00:00:00 IST 1985
Sex	male
Address	No:45,Kandy
Joining Date	Sun Dec 05 00:00:00 IST 2004

Employee Skills Details

4. If the employee wants to report an error he/she can type the error in the box & press "Send Error" button.

Human Resource Manager's role

5.If the person logged at number 1 is a human resource manager then he will be directed to `ManagerHomePage.jsp` JSP page.


New Age Employee Home - Windows Internet Explorer
http://localhost:8080/ems-war/manager/ManagerHomePage.jsp
New Age Employee Home

Human Resource Employee Management System

Manager Home M0003

Store a new Employee Record

Create a Sorted Employee Report

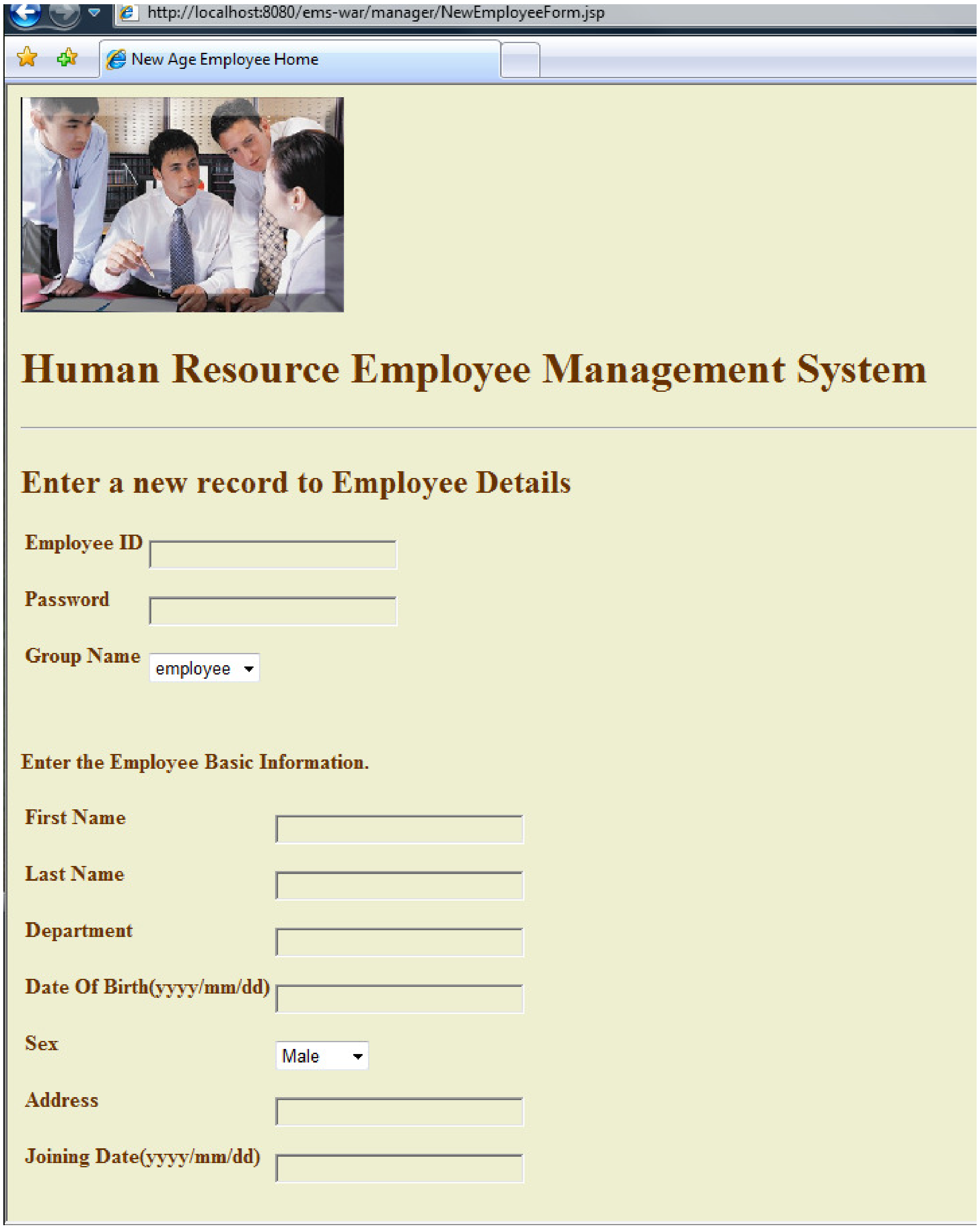
Create a Filtered Employee Report

Change Employee Details

Delete Employee Details

View Employee record errors

5. If the manager wants to store a new employee record he can click on the Store a new employee Record button. Then he/she will be directed to `NewEmployeeDataForm.jsp`. There he can enter the new employee's data & can store it.



Human Resource Employee Management System

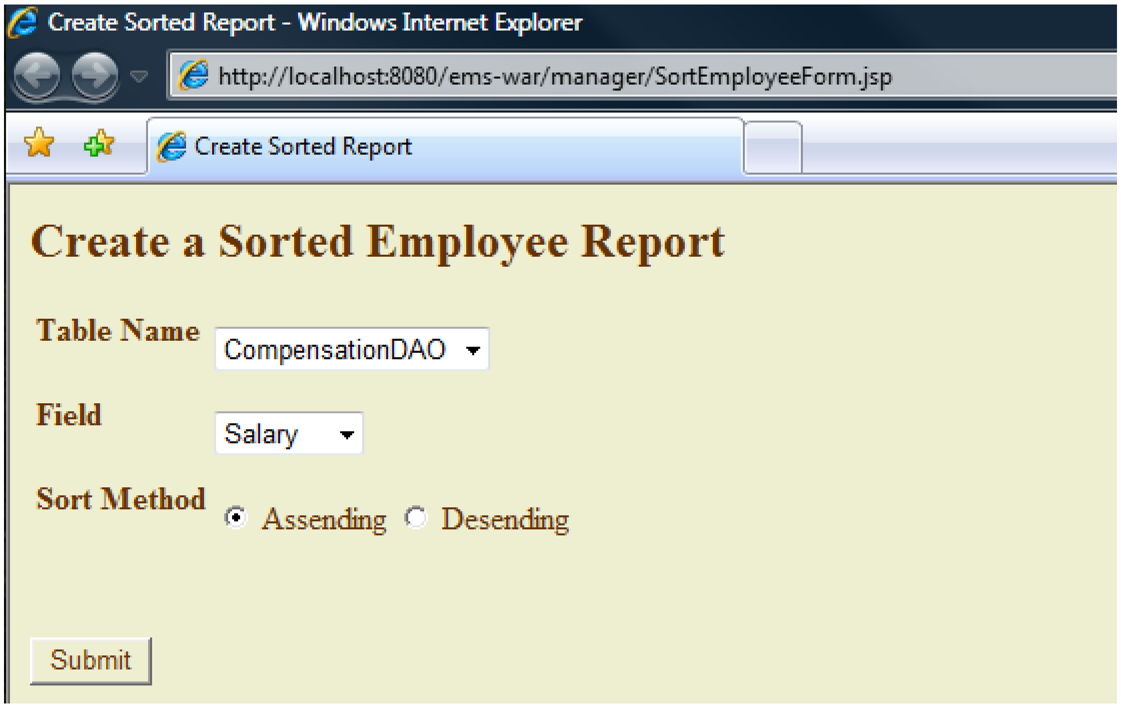
Enter a new record to Employee Details

Employee ID	
Password	
Group Name	employee

Enter the Employee Basic Information.

First Name	
Last Name	
Department	
Date Of Birth(yyyy/mm/dd)	
Sex	Male
Address	
Joining Date(yyyy/mm/dd)	

6. If the manager want to create a sorted employee report he can click on "Create a Sorted Employee Report". He will be directed to `SortEmployeeForm.jsp`.

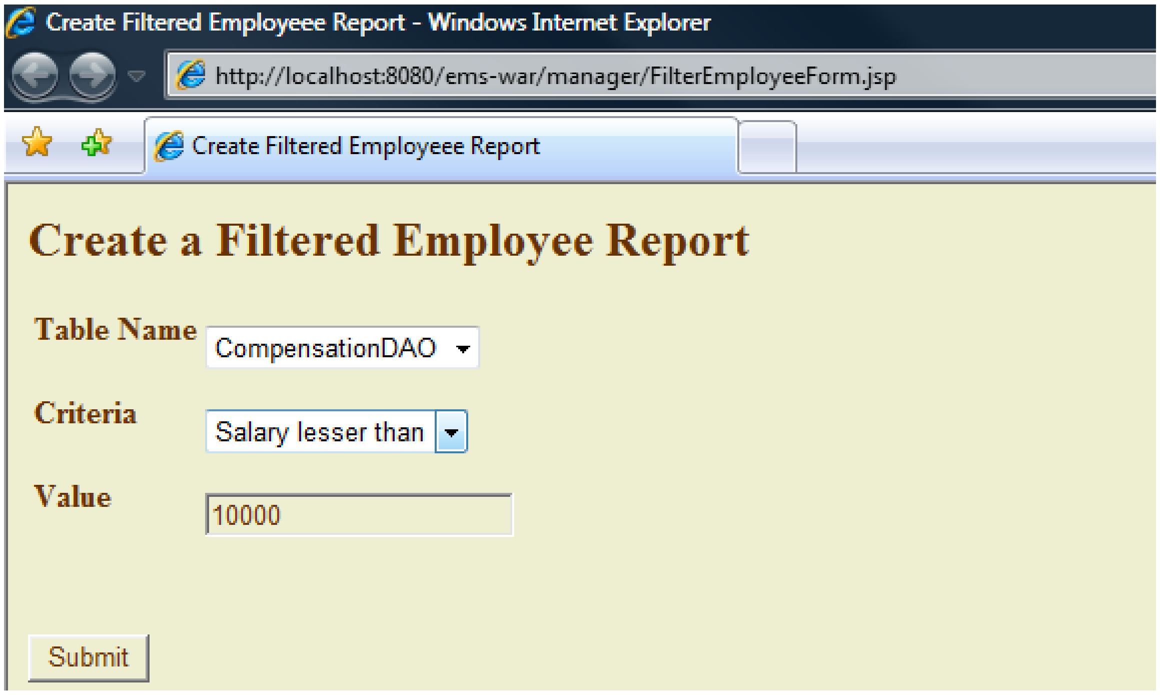


Create Sorted Report - Windows Internet Explorer
http://localhost:8080/ems-war/manager/SortEmployeeForm.jsp
Create Sorted Report

Create a Sorted Employee Report

Table Name	CompensationDAO
Field	Salary
Sort Method	[✓ Assending] Desending

7. If the manager wants to create a filtered report he can click on "Create a filtered report". He will be directed to `FilterEmployeeForm.jsp`



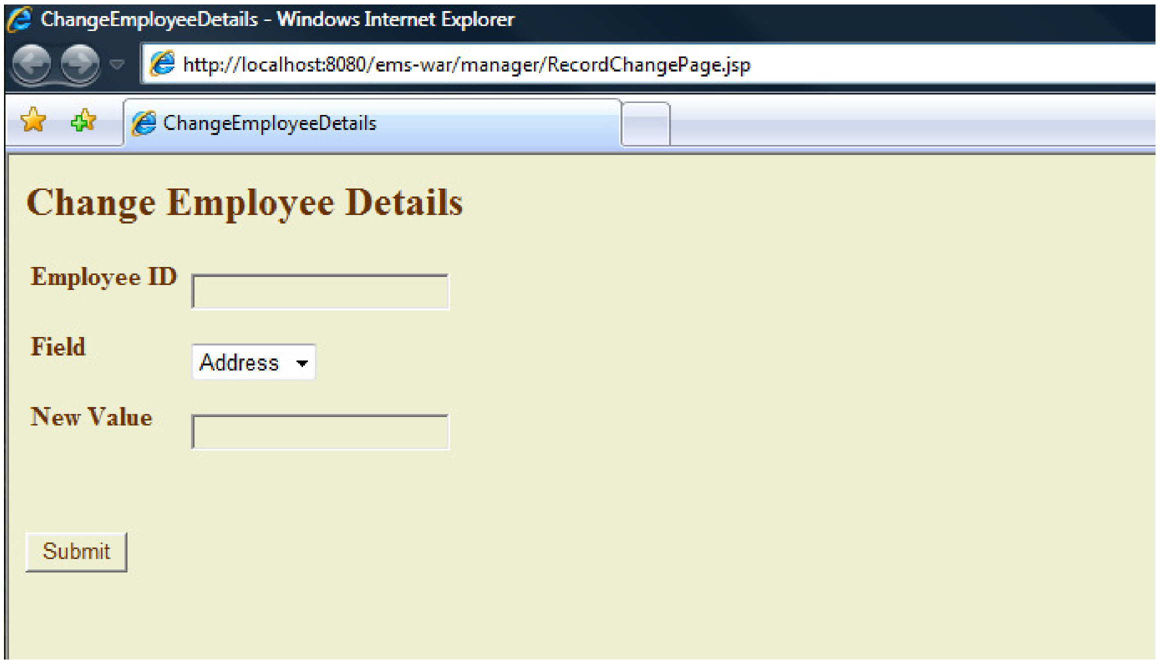
Create Filtered Employeee Report - Windows Internet Explorer
http://localhost:8080/ems-war/manager/FilterEmployeeForm.jsp
Create Filtered Employeee Report

Create a Filtered Employee Report

Table Name	CompensationDAO
Criteria	Salary lesser than
Value	10000

-

8. If the manager wants to create change some employee's data he can click on the link "Change Employee Details." Then he will be directed to `RecordChangePage.jsp`.


ChangeEmployeeDetails - Windows Internet Explorer
http://localhost:8080/ems-war/manager/RecordChangePage.jsp
ChangeEmployeeDetails

Change Employee Details

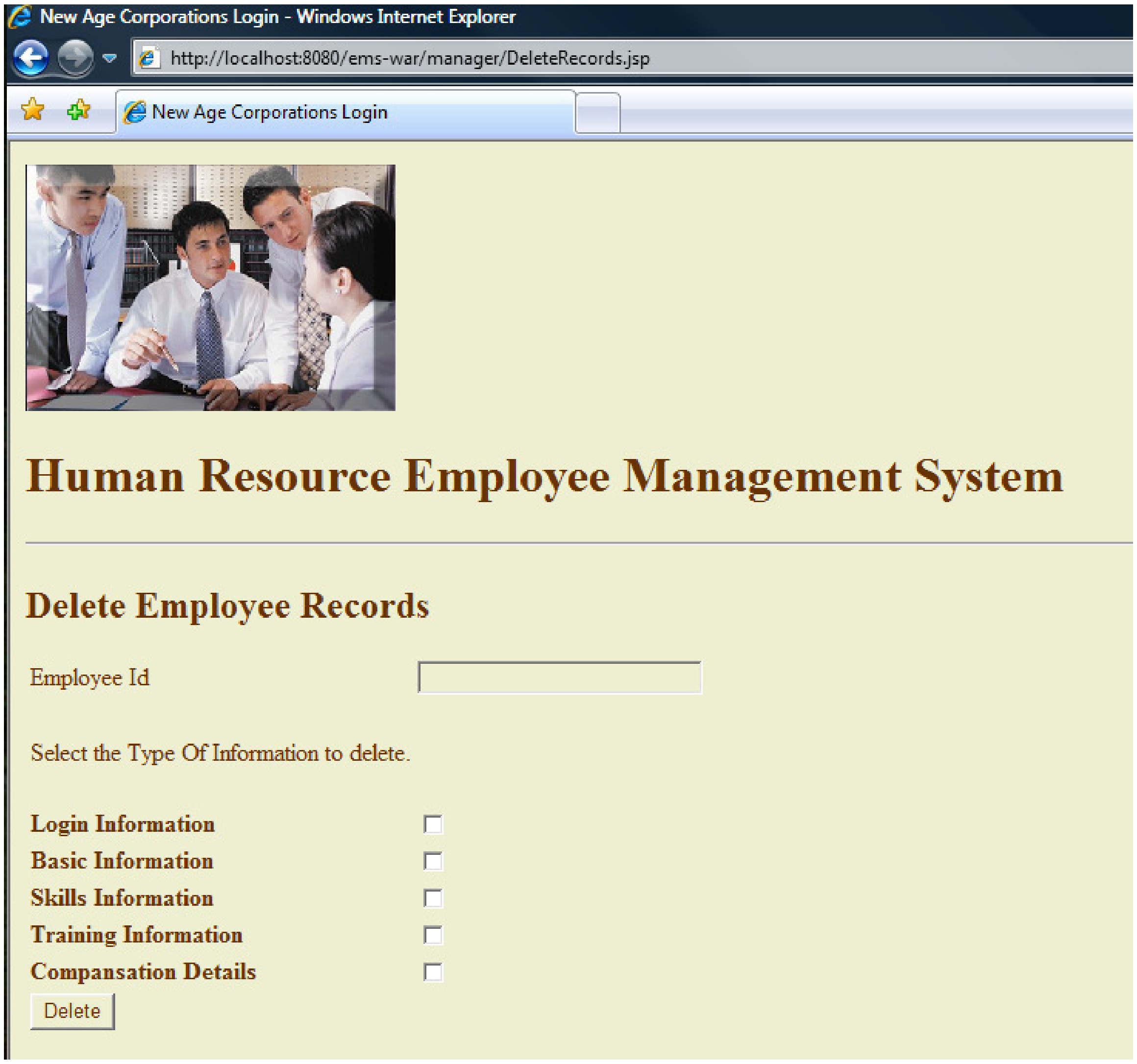
Employee ID

Field Address

New Value

Submit

9. If the manager wants to delete some employee's data he can click on the "Delete employee details" link. Then he will be directed to `deleteRecords.jsp`.


New Age Corporations Login - Windows Internet Explorer
http://localhost:8080/ems-war/manager/DeleteRecords.jsp
New Age Corporations Login

Human Resource Employee Management System

Delete Employee Records

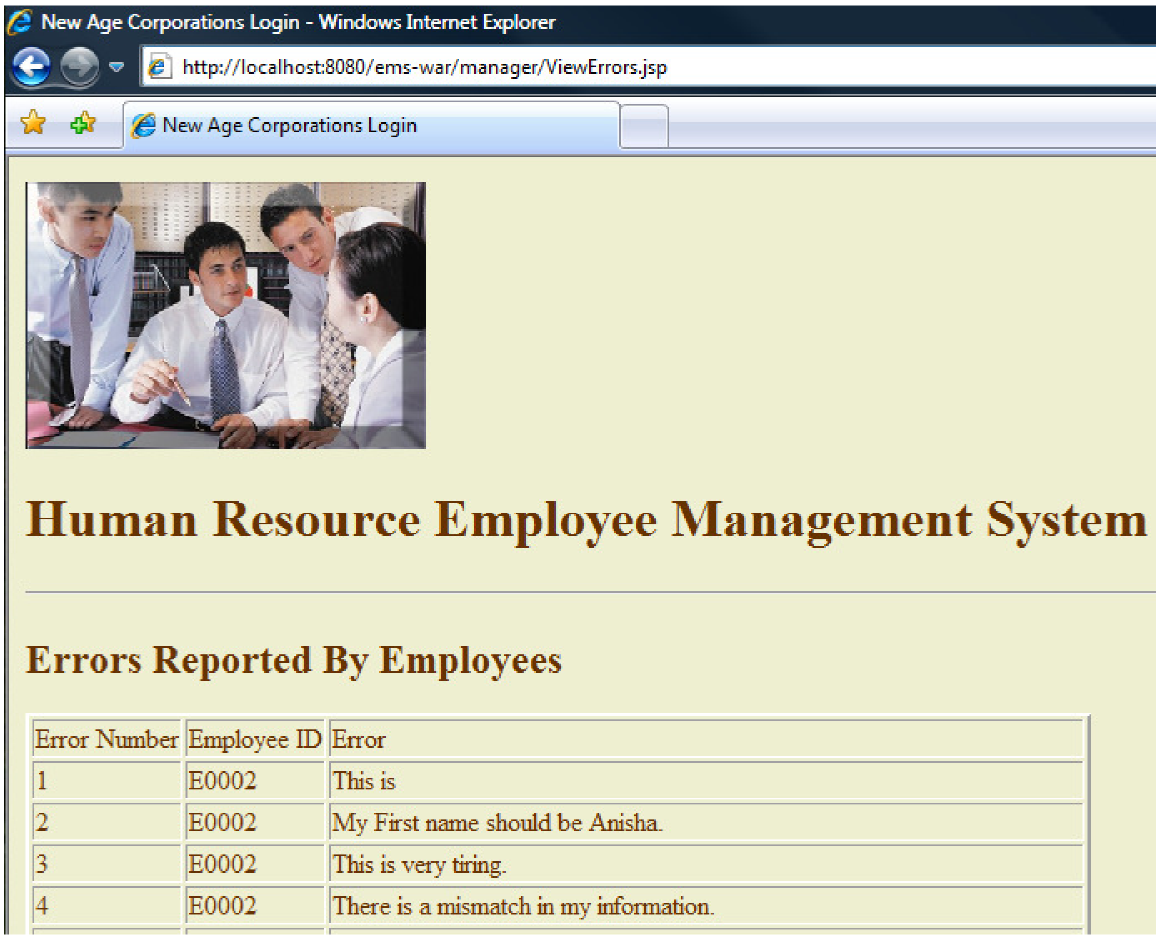
Employee Id	

Select the Type Of Information to delete.

Login Information	☐
Basic Information	☐
Skills Information	☐
Training Information	☐
Compansation Details	☐

-

10. If he wants to view the error, messages sent by employees the manager can click on View Employee Records Errors. Then it will be directed to `ViewErrors.jsp`.


New Age Corporations Login - Windows Internet Explorer
http://localhost:8080/ems-war/manager/ViewErrors.jsp
New Age Corporations Login

Human Resource Employee Management System

Errors Reported By Employees

Error Number	Employee ID	Error
1	E0002	This is
2	E0002	My First name should be Anisha.
3	E0002	This is very tiring.
4	E0002	There is a mismatch in my information.

3.3 Evaluation

To check whether the functionalities are correctly occurring I developed a couple of test cases. I am mentioning those below. For testing a got the help of a human resource manager & an employee.

Test case no:	**No: 1**
Pre Condition:	Sun J2EE Server should be running. My SQL database server should have to be started. The executer is an employee.
Description:	1. Open a web browser & in the address bar type `http://localhost:8080/ems-war/LoginForm.jsp`. 2.Enter the user name& password 3.Click on the Login button.
Expected Result:	The employee will be directed to Employee Home Page.
Actual Result:	The employee is directed to Employee Home Page.
Result:	Pass.

Test case no:	**No: 2**
Pre Condition:	Sun J2EE Server should be running. My SQL database server should have to be started. The executer is a human resource manager.
Description:	1. Open a web browser & in the address bar type `http://localhost:8080/ems-war/LoginForm.jsp`. 2.Enter the user name & password 3.Click on the Login button.
Expected Result:	The human resource manager will be directed to Manager Home Page.
Actual Result:	The human resource manager is directed to Manager Home Page.
Result:	Pass.

Test case no:	**No: 3**
Pre Condition:	Sun J2EE Server should be running. My SQL database server should have to be started. The executer is an employee. He is logged in to the system & currently he is viewing the home page.
Description:	Click on the link “view personal details” on the home page.
Expected Result:	The employee’s details will be displayed
Actual Result:	The employee’s details are displayed.
Result:	Pass.

Test case no:	**No: 4**
Pre Condition:	Sun J2EE Server should be running. My SQL database server should have to be started. In this test case both the developer (me) & a employee are participating. Employee is logged in to the system & currently he is viewing the home page. The employee initiates the test case.
Description:	1. Employee types some message in the text box that’s allocated to send error messages. 2.Employee presses the “send error” button. 3. Developer (me) logs on to the My SQL server, opens the My SQL Query browser & check whether the error is stored in the system.
Expected Result:	The error will be stored in the My SQL database.
Actual Result:	The error is stored in the My SQL database.
Result:	Pass.

Test case no:	**No: 5**
Pre Condition:	Sun J2EE Server should be running. My SQL database server should have to be started. In this test case both the developer (me) & a human resource manager are participating. Manager is logged in to the system & currently he is viewing the home page. The manager initiates the test case.
Description:	1.Human resource manager clicks on the “enter new employee record” button. 2. Human resource manager gets a form to enter a form to enter a record. 3.Human Resource manager fills the form with some imaginary data. There employee Id is filled as T0001.First name is filled as “Martin”. Other data is imaginary. 4.Developer (me) opens the My SQL Query browser & check whether the employee records are stored in the system.
Expected Result:	The employee data will be stored in the My SQL database.
Actual Result:	The employee data is stored in the My SQL database.
Result:	Pass.

Test case no:	**No: 6**
Pre Condition:	Sun J2EE Server should be running. My SQL database server should have to be started. In this test case both the developer (me) & a human resource manager are participating. Manager is logged in to the system & currently he is viewing the home page. The manager initiates the test case.
Description:	1.Human resource manager clicks on the “change employee details” button. 2.Human resource manager gets a form to enter the employee id of the employee whose details have to be changed, the field to be changed & new value. 3.Human Resource manager fills the employee ID as the previous entered ID T0001.The field to be changed is selected as first name. New value is entered as “Terry”. 4.Developer (me) opens the My SQL Query browser & check whether the first name of the employee with ID T0001 is changed in to “Terry”.
Expected Result:	The first name of the employee having ID as T0001 will be changed as Terry.
Actual Result:	The first name of the employee having ID as T0001 is changed as Terry.
Result:	Pass.

Test case no:	**No: 7**
Pre Condition:	Sun J2EE Server should be running. My SQL database server should have to be started. In this test case both the developer (me) & a human resource manager are participating. Manager is logged in to the system & currently he is viewing the home page. The manager initiates the test case.
Description:	1.Human resource manager clicks on the “delete employee details” button. 2.Human resource manager gets a form to enter the employee id of the employee whose details have to be deleted & from which tables data has to be deleted. 3.Human Resource manager fills the employee ID of the employee that data has to be deleted as the previous entered ID T0001.The table from which data has to be deleted is entered as “Basic DAO”. Then he presses submit button. 4.Developer (me) opens the My SQL Query browser & check whether that there is no basic data record exists in the database for a employee with ID “T0001”
Expected Result:	There will be no basic data record exists in the database for a employee with ID “T0001”
Actual Result:	There is no basic data record exists in the database for a employee with ID “T0001”
Result:	Pass.

Test case no:	**No: 8**
Pre Condition:	Sun J2EE Server should be running. My SQL database server should have to be started. In this test case a human resource manager is participating. Manager is logged in to the system & currently he is viewing the home page. The manager initiates the test case.
Description:	1.Human resource manager clicks on the “view Employee record errors ” button.
Expected Result:	There will be a page displaying the errors recorded by employees.
Actual Result:	A page is viewed with errors submitted by employees.
Result:	Pass.

Test case no:	**No: 9**
Pre Condition:	Sun J2EE Server should be running. My SQL database server should have to be started. A human resource manager is participating. Manager is logged in to the system & currently he is viewing the home page. The manager initiates the test case.
Description:	1.Human resource manager clicks on the "create a Sorted Employee Report" button. 2.Human resource manager gets a form to enter a table name to sort, the data field to sort, the order of sorting. He enters the table name as "BasicDAO", sort variable as First Name, order as as descending. 3. Then he presses "Submit". 4. After that he gets a sorted report. Manually he checks whether the report is sorted by descending order of first name.
Expected Result:	The data in the report will be sorted by descending order of first name.
Actual Result:	The data in the report will be sorted by descending order of first name.
Result:	Pass.

Test case no:	**No: 10**
Pre Condition:	Sun J2EE Server should be running. My SQL database server should have to be started. In this test case both the developer (me) & a human resource manager are participating. Manager is logged in to the system & currently he is viewing the home page. The manager initiates the test case.
Description:	1.Human resource manager clicks on the "create a Filtered Employee Report" button. 2.Human resource manager gets a form to enter the table name used to be filtered, the criteria to be filtered & value. 3.Human Resource manager fills the table name as CompensationDAO .The criteria is salary greater than. Value is entered as 10000. 4. A report will be generated. 4.Developer (me) opens the My SQL Query browser & check whether the report contains information only having salary greater than 10000.
Expected Result:	There will be no information of the employees having lesser than 10000 salary in the created report.
Actual Result:	There is no information of the employees having lesser than 10000 salary in the created report.
Result:	Pass.

4. Review Stage Of Development

4.1Current Stage of Development

Currently my proposed prototype solution has almost all the essential features that should be functioning. I am mentioning those below one by one.

- When a new employee joins human resource mangers can enter his/her all the details to the database. If the employee is a trainee his trainee information is also entered to the database.
- When a human resource mangers want to change some employee data they can change what they require just by one or two clicks.
- When an employee leaves the company & if the company doesn't need to store that employee's information after certain period human resource managers can delete all records of that employee easily.
- Employee managers can create the filtered reports about employees satisfying certain criteria within just one or two clicks using one table at a time.
- Employee managers can create the sorted reports about employees using one table at a time.
- Employees also can see their details stored in the system by just one or two clicks without filling different types of forms.
- If the employee information changes or employee thinks that their information is wrong they can inform the human resource managers just by typing the error & pressing OK.
- Human Resource Managers can view the errors reported by the employees easily.

4.2 Limitations at this stage

Further more I find that some features of my solution that has limitations. I am mentioning those below here.

- The database I am using for the proposed software prototype is not well organized. Tables are not organized properly. Some times the already selected primary keys foreign keys are not the best possible primary keys & foreign keys.

- The current security level doesn't seem to be enough since the database contains all information in plain text. Further more when data is moving in the intranet also data is not encrypted.

- In my software prototype the human resource managers can't differentiate the read errors & unread errors.

- The human resource mangers can currently create reports of the employee by querying one table at a time only. If some reports need to aggregate the data from two or three tables then there exists a problem.

4.3 Additional Work Need To Be Done

To overcome the above-mentioned limitations I propose the below steps to be taken.

- The database should be redesigned according to current needs. Appropriate indexes have to be created for faster searching facilities. Referential integrity has to be maintained correctly. Table relationships, validation rules have to be taken in to consideration when designing the database.

- At least the most confidential data should be made encrypted in the database. It's better if we can make the data flowing in the intranet encrypted.

- The ability to differentiate read error messages & unread messages can be implemented by just adding another column to the table containing errors. That column can accept a Boolean value to mark the message as read or unread.

- The searching facility should be further widened to allow searching to be done using multiple tables once. This can be done with the help of EJB Query Language.

5. Review project management

5.1 Review the effectiveness of the project management life cycle

I used spiral model one of the iterative life cycle models as the project management life cycle. Each iteration in this life cycle consists of six phases as I am mentioning below.

Phase	Activities
Customer Communication	Continuous communication with the customer to understand system requirements
Planning	Schedule estimation, resource estimation
Risk Analysis	Identify, estimate, and manage technical risks as well as management risks
Engineering	Requirement gathering & designing
Construction	Coding, testing, deploying the software in a customer's place
Customer evaluation	Customer evaluates the software & feedback is implemented to the next iteration.

At first I have to mention when I was in the mid of developing the solution a requirement evolved, the employee should have a efficient mechanism to inform the human resource managers if they find that their records contain errors or they think that their records should be changed. Since I was using this spiral model I could consider that in the iteration after the iteration I was continuing at that time. This was very effective for me to consider newly evolved requirement since if I had selected a life cycle where all the requirements are first identified & then designing is done I would have been in a real problem.

After identifying requirements I gave priorities to them. Requirements were taken in to consideration according to the priority level. So I was able to implement the highest priority requirements earlier & show the prototype to my clients (human resource managers & employees) & get an early feedback about most important requirements. That was another advantage I got. Further after each & every iteration customer evaluation was done. So that I got valuable feedback about my solution prototype at each iteration whether they were satisfied with what I was developing. All of these things I got due to the spiral model.

I implemented the delete employee records functionality. After evaluating the prototype my clients were not satisfied. They told me that when the delete button was pressed all the records about that particular employee was getting deleted. Then they said they wanted an option to select from which table the data should be deleted. So in the next iteration I implemented it according to their wish. So at the early ages itself I got their view. If I had developed further without knowing their ideas changing it later would be time consuming. It was stopped due to my life cycle model.

Since I was communicating with my clients at the start of each new iteration their requirements became very clear to me. I think that no ambiguity was found in requirements due to the extra ordinary client communication in my life cycle model.

With this spiral model I had the opportunity to start the designing earlier. That also one of the advantages I got. Further the risk analysis was done in each iteration so I understood when I was running out of time. So I managed my schedules according to time limit. I didn't have much knowledge about coding parts in message driven beans & sending messages to Destinations. It was a technical risk. Due to this life cycle I managed my schedule so that I could learn the coding & implement the functionality within the time limit.

Due to those reasons I think that I had selected the most suitable life cycle model for my proposed prototype solution.

5.2 Resources used for the proposed software prototype

- **Data Loader Version 3.6**: This software was used to convert flat data files & Microsoft Access database in to MySQL database.

- **MySQL Delete (Remove) Duplicate Entries 7.0**: This was used to remove the duplicate entries in the created MySQL database.

- **NetBeans IDE**: Developing all the coding parts (developing entity classes, session beans, Message driven beans, servlet classes, part of the designing of Java Server Pages) were done with the help of this software.

- **Sun Java System Application Server Platform Edition 9.0_01**. : Deploying my prototype solution was done on this server.

- **Jude Professional:** Drawing UML diagrams was done with the help of this software.

- **Macromedia Dreamviewer:** Designing some of the Java Server pages were done with the help of this software.

6. Review of personal development

This was the first time I did a complete software development activities requirement gathering, requirement analysis, design, coding, testing all myself in a real life scenario. Prior to this project I had done some projects where I did only the design & coding phases.

When requirement gathering I had to prepare a questioners to gather the requirements. It was also a rewarding experience.

This is the first time I developed a project according to a project management life cycle. So this was a very good experience. I understood that how to schedule the activities, how to analyze the risks, especially to technical risk since I was somewhat amateur to message driven beans.

The other thing is modeling the J2EE application. This is the first time I am doing it. I actually enjoyed drawing UML diagrams to the system. I got the real life experience in developing a project.

When considering coding I learnt lot of Java EE concepts especially message driven beans, servlets, Java Server pages, session beans, EJB Security.

<u>Appendix 1</u>

Interview Plan

There was five parts in the interview.: Introduction, warm-up, main session, cool-off period, closing session. I am mentioning below the questions used in main session.

1. Have you used Internet?
2. Have you used Internet based mail systems (Yahoo mail,Google mail, Hot mail)?
3. What's your attitude about current employee management procedure in the company?
4. Are you satisfied with the way employee report generation is done?
5. What are the problems in report generation?
6. What kind of reports has to be generated?
7. What are the improvements you would like to have in the current employee management system?

<u>Appendix 2</u>

Questionnaire

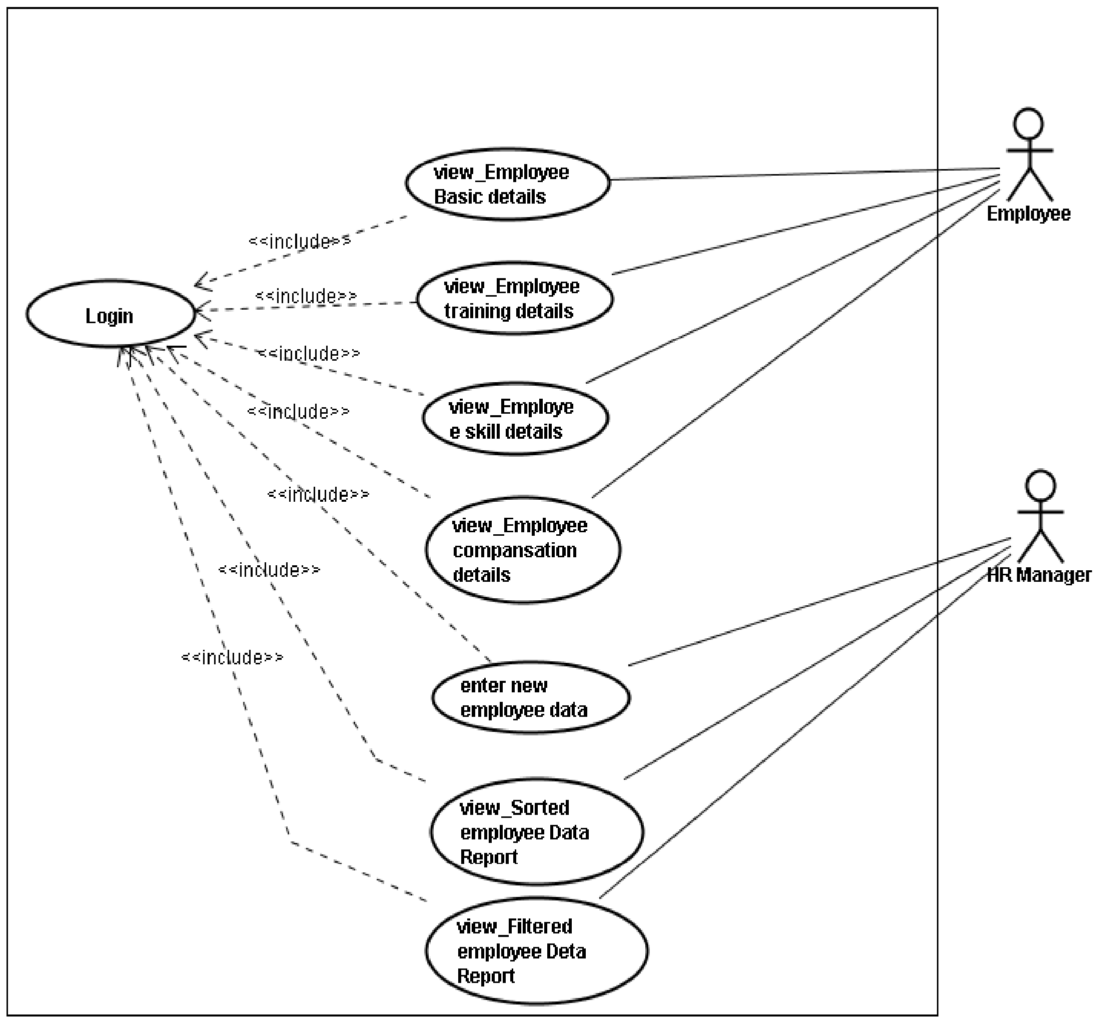
These were the questions on questioner.

1. Have you used a computer?
2. How often do you use the Internet?
3. Have you used Internet based mail systems (Yahoo mail, Google mail, Hot mail)?

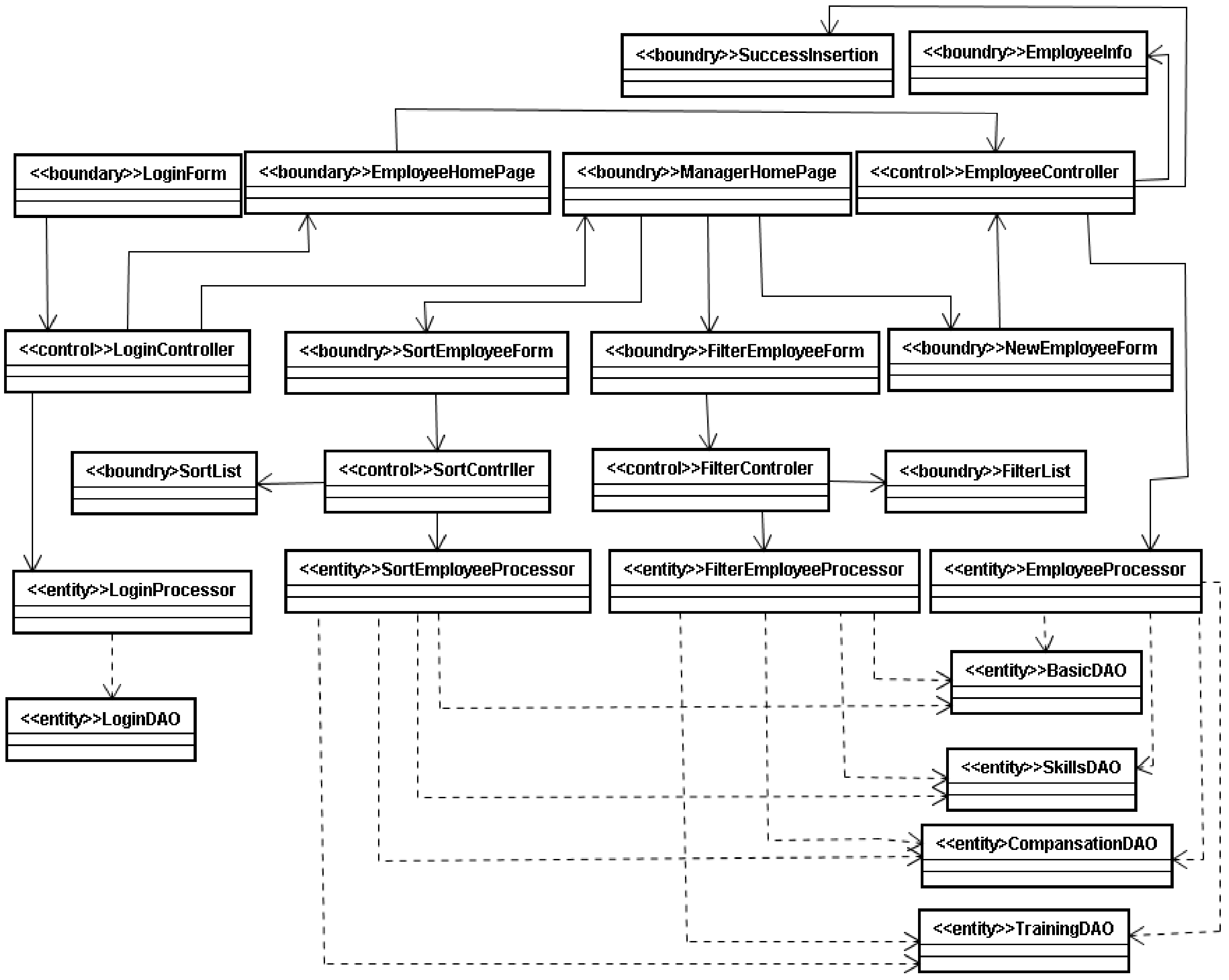
4. Do you think the current employee management system is convenient for you?
5. Do you like to view your data electronically?
6. Is there currently any automated system to get your data?
7. Are you satisfied with the company's employee management system?

-

Appendix 3


view_Employee Basic details
view_Employee training details
Login
view_Employe e skill details
view_Employee compansation details
enter new employee data
view_Sorted employee Data Report
view_Filtered employee Deta Report
<<include>>
Employee
HR Manager

Appendix4


<<boundry>>SuccessInsertion
<<boundry>>EmployeeInfo
<<boundary>>LoginForm
<<boundary>>EmployeeHomePage
<<boundry>>ManagerHomePage
<<control>>EmployeeController
<<control>>LoginController
<<boundry>>SortEmployeeForm
<<boundry>>FilterEmployeeForm
<<boundry>>NewEmployeeForm
<<boundry>SortList
<<control>>SortContrller
<<control>>FilterControler
<<boundry>>FilterList
<<entity>>LoginProcessor
<<entity>>SortEmployeeProcessor
<<entity>>FilterEmployeeProcessor
<<entity>>EmployeeProcessor
<<entity>>BasicDAO
<<entity>>LoginDAO
<<entity>>SkillsDAO
<<entity>CompansationDAO
<<entity>>TrainingDAO

-

Appendix 5
// BasicDAO.java

```
package Employee;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

/**
 * Entity class BasicDAO
 *
 * @author Melroni
 */
@Entity
@Table(name = "basic_details")
@NamedQueries( {
        @NamedQuery(name = "BasicDAO.findByEmpId", query =
"SELECT b FROM BasicDAO b WHERE b.empId = :empId"),
        @NamedQuery(name = "BasicDAO.findByFname", query =
"SELECT b FROM BasicDAO b WHERE b.fname = :fname"),
        @NamedQuery(name = "BasicDAO.findByLname", query =
"SELECT b FROM BasicDAO b WHERE b.lname = :lname"),
        @NamedQuery(name = "BasicDAO.findByDepartment", query =
"SELECT b FROM BasicDAO b WHERE b.department =
:department"),
        @NamedQuery(name = "BasicDAO.findByDateOfBirth", query =
"SELECT b FROM BasicDAO b WHERE b.dateOfBirth =
:dateOfBirth"),
        @NamedQuery(name = "BasicDAO.findBySex", query = "SELECT
b FROM BasicDAO b WHERE b.sex = :sex"),
        @NamedQuery(name = "BasicDAO.findByAddress", query =
"SELECT b FROM BasicDAO b WHERE b.address = :address"),
        @NamedQuery(name = "BasicDAO.findByJoiningdate", query =
"SELECT b FROM BasicDAO b WHERE b.joiningdate = :joiningdate"),
        @NamedQuery(name = "BasicDAO.findHottestQuery", query =
"SELECT b FROM BasicDAO b ")

    })
public class BasicDAO implements Serializable {

    @Id
    @Column(name = "emp_id", nullable = false)
    private String empId;

    @Column(name = "fname", nullable = false)
    private String fname;

    @Column(name = "lname", nullable = false)
    private String lname;

    @Column(name = "department", nullable = false)
    private String department;

    @Column(name = "date_of_birth", nullable = false)
    @Temporal(TemporalType.TIMESTAMP)
    private Date dateOfBirth;

    @Column(name = "sex", nullable = false)
    private String sex;

    @Column(name = "address", nullable = false)
    private String address;

    @Column(name = "Joining_date", nullable = false)
    @Temporal(TemporalType.TIMESTAMP)
    private Date joiningdate;

    /** Creates a new instance of BasicDAO */
    public BasicDAO() {
    }
    /**
     * Creates a new instance of BasicDAO with the specified values.
     * @param empId the empId of the BasicDAO
     */
    public BasicDAO(String empId) {
        this.empId = empId;
    }

    /**
     * Creates a new instance of BasicDAO with the specified values.
     * @param empId the empId of the BasicDAO
     * @param fname the fname of the BasicDAO
     * @param lname the lname of the BasicDAO
     * @param department the department of the BasicDAO
     * @param dateOfBirth the dateOfBirth of the BasicDAO
     * @param sex the sex of the BasicDAO
     * @param address the address of the BasicDAO
     * @param joiningdate the joiningdate of the BasicDAO
     */
    public BasicDAO(String empId, String fname, String lname, String
department, Date dateOfBirth, String sex, String address, Date
joiningdate) {
        this.empId = empId;
        this.fname = fname;
        this.lname = lname;
        this.department = department;
        this.dateOfBirth = dateOfBirth;
        this.sex = sex;
        this.address = address;
        this.joiningdate = joiningdate;
    }

    /**
     * Gets the empId of this BasicDAO.
     * @return the empId
     */
    public String getEmpId() {
        return empId;
    }
```

```
/**
 * Sets the empId of this BasicDAO to the specified value.
 * @param empId the new empId
 */
public void setEmpId(String empId) {
   this.empId = empId;
}

/**
 * Gets the fname of this BasicDAO.
 * @return the fname
 */
public String getFname() {
   return this.fname;
}

/**
 * Sets the fname of this BasicDAO to the specified value.
 * @param fname the new fname
 */
public void setFname(String fname) {
   this.fname = fname;
}

/**
 * Gets the lname of this BasicDAO.
 * @return the lname
 */
public String getLname() {
   return this.lname;
}

/**
 * Sets the lname of this BasicDAO to the specified value.
 * @param lname the new lname
 */
public void setLname(String lname) {
   this.lname = lname;
}
```

```
/**
 * Gets the department of this BasicDAO.
 * @return the department
 */
public String getDepartment() {
   return this.department;
}

/**
 * Sets the department of this BasicDAO to the specified value.
 * @param department the new department
 */
public void setDepartment(String department) {
   this.department = department;
}

/**
 * Gets the dateOfBirth of this BasicDAO.
 * @return the dateOfBirth
 */
public Date getDateOfBirth() {
   return this.dateOfBirth;
}

/**
 * Sets the dateOfBirth of this BasicDAO to the specified value.
 * @param dateOfBirth the new dateOfBirth
 */
public void setDateOfBirth(Date dateOfBirth) {
   this.dateOfBirth = dateOfBirth;
}

/**
 * Gets the sex of this BasicDAO.
 * @return the sex
 */
public String getSex() {
   return this.sex;
}
```

```
    /**
     * Sets the sex of this BasicDAO to the specified value.
     * @param sex the new sex
     */
    public void setSex(String sex) {
        this.sex = sex;
    }

    /**
     * Gets the address of this BasicDAO.
     * @return the address
     */
    public String getAddress() {
        return this.address;
    }

    /**
     * Sets the address of this BasicDAO to the specified value.
     * @param address the new address
     */
    public void setAddress(String address) {
        this.address = address;
    }

    /**
     * Gets the joiningdate of this BasicDAO.
     * @return the joiningdate
     */
    public Date getJoiningdate() {
        return this.joiningdate;
    }
    public String[] getFields()
    {
        String
temporary[]={this.empId,this.getFname(),this.getLname(),this.getDepart
ment(),this.getDateOfBirth().toString(),this.getSex(),this.getAddress(),this.
getJoiningdate().toString()};
        return temporary;
    }
```

-

```
    /**
     * Sets the joiningdate of this BasicDAO to the specified value.
     * @param joiningdate the new joiningdate
     */
    public void setJoiningdate(Date joiningdate) {
        this.joiningdate = joiningdate;
    }

    /**
     * Returns a hash code value for the object.  This implementation
computes
     * a hash code value based on the id fields in this object.
     * @return a hash code value for this object.
     */



    /**
     * Determines whether another object is equal to this BasicDAO.  The
result is
     * <code>true</code> if and only if the argument is not null and is a
BasicDAO object that
     * has the same id field values as this object.
     * @param object the reference object with which to compare
     * @return <code>true</code> if this object is the same as the
argument;
     * <code>false</code> otherwise.
     */



    /**
     * Returns a string representation of the object.  This implementation
constructs
     * that representation based on the id fields.
     * @return a string representation of the object.
     */
    @Override
    public String toString() {
        return "enterpriseComponents.entity.BasicDAO[empId=" + empId +
"]";
    }
```

```
}
```

Appendix 6

```
package Manager;

import javax.ejb.Remote;
import java.util.List;

/**
 * This is the business interface for FilterEmployeeProcessor enterprise
bean.
 */
@Remote
public interface FilterEmployeeProcessorRemote {

    public void deleteLoginDAOS(String empId);


    public void deleteCompansationDAO(String empId);


    public void deleteTrainingDAO(String empId);


    public void deleteSkillsDAO(String empId);


    public void deleteBasicDAO(String empId);

    public List filterBasicDAO(String option,String value);
}



/*
 * FilterEmployeeProcessorBean.java*/
```

```
package Manager;

import javax.ejb.Stateless;
import javax.persistence.*;
import enterpriseComponents.entity.*;
import javax.annotation.security.RolesAllowed;
import java.util.List;

/**
 *
 * @author Melroni
 */
@Stateless
public class FilterEmployeeProcessor implements
enterpriseComponents.session.FilterEmployeeProcessorRemote {
   @PersistenceContext(unitName="ems-ejbPU")
   private EntityManager manager;

   @RolesAllowed("managers")
   public void deleteBasicDAO(String empId)
   {
      BasicDAO object=manager.find(BasicDAO.class,empId);
      if(object!=null)
      manager.remove(object);
   }

   @RolesAllowed("managers")
   public void deleteSkillsDAO(String empId)
   {
      SkillsDAO object=manager.find(SkillsDAO.class,empId);
      if(object!=null)
      manager.remove(object);
   }

   @RolesAllowed("managers")
   public void deleteTrainingDAO(String empId)
   {
      TrainingDAO object=manager.find(TrainingDAO.class,empId);
      if(object!=null)
      manager.remove(object);
```

```
    }

    @RolesAllowed("managers")
    public void deleteCompansationDAO(String empId)
    {
        CompansationDAO
object=manager.find(CompansationDAO.class,empId);
        if(object!=null)
        manager.remove(object);
    }

    @RolesAllowed("managers")
    public void deleteLoginDAOS(String empId)
    {
        UserDAO objectOne=manager.find(UserDAO.class,empId);
        GroupDAO objectTwo=manager.find(GroupDAO.class,empId);
        if(objectOne!=null && objectTwo!=null)
        {
            manager.remove(objectOne);
            manager.remove(objectTwo);
        }
    }
    public List filterBasicDAO(String option,String value)
    {
        Query
query=manager.createNamedQuery("BasicDAO.findByEmpId");
        query.setParameter("empId",value);
        List<BasicDAO> objects=(List<BasicDAO>)query.getResultList();
        return objects;
    }


}
```

-

Appendix 7

```
package error;

import javax.ejb.ActivationConfigProperty;
import javax.ejb.MessageDriven;
import javax.jms.*;
import javax.jms.MessageListener;
import javax.persistence.*;
import enterpriseComponents.entity.ErrorDAO;
/**
 * Entity class ErrorSender
 *
 * @author Melroni
 */
@MessageDriven(mappedName = "jms/ErrorSender", activationConfig =
{
      @ActivationConfigProperty(propertyName = "acknowledgeMode",
propertyValue = "Auto-acknowledge"),
      @ActivationConfigProperty(propertyName = "destinationType",
propertyValue = "javax.jms.Queue")
   })
public class ErrorSender implements MessageListener {
   @PersistenceContext(unitName="ems-ejbPU")
   private EntityManager manager;
   static int counter=0;
   /** Creates a new instance of ErrorSender */
   public ErrorSender() {
   }

   public void onMessage(Message msg)
   {
      try
                                          {
                                                   if(msg instanceof TextMessage)
                                                                              {
                                                                                    TextMessag
                        counter++;
                        ErrorDAO object=new
ErrorDAO(counter,(String)txtMsg.getObjectProperty("id"),txtMsg.getTe
xt());
                        manager.persist(object);
                                                                                                      }
                                                                                                  else
                                                                                                      {
                                                                                                                System.ou
                                                                                                      }
       }
                                  catch(JMSException e)
                                                {
                                                                         e.printStackTrace();
                                                }
                                    catch(Throwable te)
                                                {
                                                                        te.printStackTrace();
                                                }
   }
}


/*

 * ErrorController.java

 *

 * Created on December 17, 2009, 12:39 PM

 */


package Error;


import java.io.*;

import java.net.*;

import javax.jms.*;
import javax.naming.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.ejb.*;

import java.util.List;

import enterpriseComponents.entity.ErrorDAO;

/**

 *

 * @author Melroni

 * @version

 */

public class ErrorController extends HttpServlet {

            InitialContext       jndiContext=null;

            ConnectionFactory conFact=null;

         Connection          con=null;

         Session          session=null;

         Destination          dest=null;

         MessageProducer  msgPrd=null;

         TextMessage       msg=null;

   /** Processes requests for both HTTP <code>GET</code> and <code>POST</code>
methods.

    * @param request servlet request

    * @param response servlet response

    */

   @EJB

   private enterpriseComponents.session.ErrorRetrieverRemote errorRetreiver;
```

```
  protected void processRequest(HttpServletRequest request, HttpServletResponse
response)

  throws ServletException, IOException

  {

    if(request.isUserInRole("employees"))

     send(request,response);

    else

    {

       List<ErrorDAO> objects=errorRetreiver.getErrors();

       HttpSession session=request.getSession();

       session.setAttribute("errors",objects);

       RequestDispatcher
toDisplay=getServletContext().getRequestDispatcher("/manager/viewErrors.jsp");

       toDisplay.forward(request,response);

    }


  }

  public void send(HttpServletRequest request, HttpServletResponse response)throws
ServletException,IOException

  {

     String error=request.getParameter("error");

     String id=(String)request.getSession().getAttribute("empId");


     try

       {

           jndiContext=new InitialContext();

       }
```

```
        catch(NamingException e)

        {

            System.out.println("Could not create JNDI"+"context:"+e.toString());

        }

        try

        {

            conFact=(ConnectionFactory)jndiContext.lookup("jms/ErrorSenderFactory");

            dest=(Queue)jndiContext.lookup("jms/ErrorSender");

        }

        catch(NamingException e)

        {

                System.out.println("Could not find the object:"+e.toString());

        }

        try

        {

            con=conFact.createConnection();

            session=con.createSession(false,Session.AUTO_ACKNOWLEDGE);

            msgPrd=session.createProducer(dest);

            msg=session.createTextMessage();

                msg.setObjectProperty("id",id);

            msg.setText(error);

                msgPrd.send(msg);

                RequestDispatcher
toDisplay=getServletContext().getRequestDispatcher("/employee/ErrorSendConfirmatio
n.jsp");

                toDisplay.forward(request,response);
        }
        catch(JMSException e)
        {
            System.out.println("Exception: "+e.toString());
        }
        finally
        {
            if(con!=null)
            {
                try
                {
                    con.close();
                }
                catch(JMSException e){}
            }
        }
    }
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /** Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }
```

-

```
    /** Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /** Returns a short description of the servlet.
     */
    public String getServletInfo() {
        return "Short description";
    }
    // </editor-fold>
}
```

Appendix 8

```
/*

 * LoginController.java

 *

 * Created on December 15, 2009, 10:59 AM

 */


package Login;


import java.io.*;

import java.net.*;


import javax.servlet.*;

import javax.servlet.http.*;


/**

 *

 * @author Melroni

 * @version

 */

public class LoginController extends HttpServlet {


   /** Processes requests for both HTTP <code>GET</code> and <code>POST</code>
methods.

    * @param request servlet request

    * @param response servlet response

    */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)

    throws ServletException, IOException {

        HttpSession session=request.getSession(true);

        session.setAttribute("empId",request.getUserPrincipal().getName());



        if(request.isUserInRole("managers"))

            response.sendRedirect("http://localhost:8080/ems-
war/manager/ManagerHomePage.jsp");

        else

            response.sendRedirect("http://localhost:8080/ems-
war/employee/EmployeeHomePage.jsp");



    }



    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">

    /** Handles the HTTP <code>GET</code> method.

     * @param request servlet request

     * @param response servlet response

     */

    protected void doGet(HttpServletRequest request, HttpServletResponse response)

    throws ServletException, IOException {

        processRequest(request, response);

    }



    /** Handles the HTTP <code>POST</code> method.

     * @param request servlet request
     * @param response servlet response
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /** Returns a short description of the servlet.
     */
    public String getServletInfo() {
        return "Short description";
    }
    // </editor-fold>
}
```

-

Appendix 9

```
/*

 * EmployeeController.java

 *

 * Created on December 16, 2009, 2:09 PM

 */


package Employee;


import enterpriseComponents.session.EmployeeProcessorRemote;

import java.io.*;

import java.net.*;


import javax.servlet.*;

import javax.servlet.http.*;

import javax.ejb.*;

import java.util.Date;

/**

 *

 * @author Melroni

 * @version

 */

public class EmployeeController extends HttpServlet {


   /** Processes requests for both HTTP <code>GET</code> and <code>POST</code>
methods.

     * @param request servlet request
```

```
    * @param response servlet response
    */
   @EJB
   private enterpriseComponents.session.EmployeeProcessorRemote employeeProcessor;
   protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
       if(request.isUserInRole("employees"))
       {
             HttpSession session=request.getSession();
             String id=(String)session.getAttribute("empId");


             String basicinfo[]=employeeProcessor.getBasicDAO(id);
             String compansationInfo[]=employeeProcessor.getCompansationDAO(id);
             String skillsInfo[]=employeeProcessor.getSkillsDAO(id);
             String trainigInfo[]=employeeProcessor.getTrainingDAO(id);
             if(basicinfo!=null)
                session.setAttribute("basic",basicinfo);
             if(compansationInfo!=null)
                session.setAttribute("compansation",compansationInfo);
             if(skillsInfo!=null)
                session.setAttribute("skills",skillsInfo);
             if(trainigInfo!=null)
             session.setAttribute("training",trainigInfo);
             RequestDispatcher
toDisplay=request.getRequestDispatcher("/employee/ViewEmployeeDetails.jsp");
             toDisplay.forward(request,response);
       }

       else

       {

             String empId=request.getParameter("empId");

             String Password=request.getParameter("Password");

             String group=request.getParameter("group");


             String fname=request.getParameter("fname");

             String lname=request.getParameter("lname");

             String department=request.getParameter("department");

             String DateOfBirth=request.getParameter("DateOfBirth");

             String sex=request.getParameter("sex");

             String address=request.getParameter("address");

             String joiningDate=request.getParameter("joiningDate");


             if((!empId.isEmpty()) && (!Password.isEmpty()))

                employeeProcessor.enterUserDAO(empId,Password);

             if((!empId.isEmpty()) && (!group.isEmpty()))

                employeeProcessor.enterGroupDAO(empId,group);

             if((!empId.isEmpty()) && (!fname.isEmpty()) && (!lname.isEmpty()) &&
(!department.isEmpty()) && (!DateOfBirth.isEmpty())&& (!sex.isEmpty()) &&
(!address.isEmpty()) && (!joiningDate.isEmpty()))


employeeProcessor.enterBasicDAO(empId,fname,lname,department,DateOfBirth,sex,ad
dress,joiningDate);


             String Qualification=request.getParameter("Qualification");

             String experience=request.getParameter("experience");
```

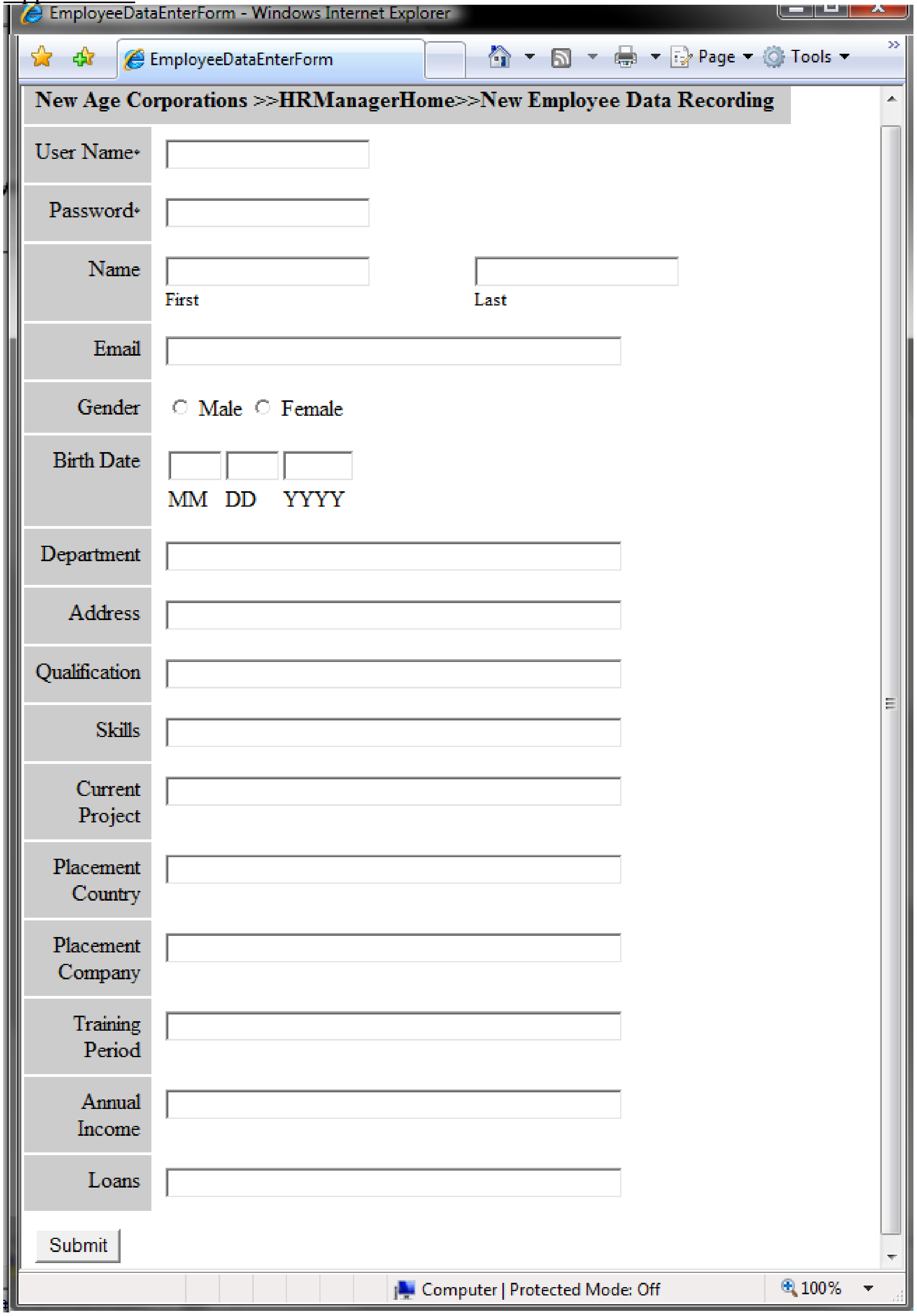
```
            String skill=request.getParameter("skill");


            if((!empId.isEmpty())  &&  (!Qualification.isEmpty())  &&
(!experience.isEmpty())&& (!skill.isEmpty()))

               employeeProcessor.enterSkillsDAO(empId,Qualification,experience,skill);


            String ProjectIn=request.getParameter("ProjectIn");

            String sdate=request.getParameter("sdate");

            String edate=request.getParameter("edate");


            if((!empId.isEmpty()) && (!ProjectIn.isEmpty()) &&(!sdate.isEmpty()) &&
(!edate.isEmpty()))

               employeeProcessor.enterTrainingDAO(empId,ProjectIn,sdate,edate);


            String Salary=request.getParameter("Salary");

            String mIncome=request.getParameter("mIncome");

            String loanType=request.getParameter("loanType");

            String mPremium=request.getParameter("mPremium");

            String taxPercentage=request.getParameter("taxPercentage");


            if((!empId.isEmpty()) && (!Salary.isEmpty()) && (!mIncome.isEmpty()) &&
(!loanType.isEmpty()) && (!mPremium.isEmpty())  && (!taxPercentage.isEmpty()))


employeeProcessor.enterCompansationDAO(empId,Salary,mIncome,loanType,mPremiu
m,taxPercentage);
```

```
            RequestDispatcher
toDisplay=getServletContext().getRequestDispatcher("/manager/SuccessfullDataInsertio
n.jsp");

            toDisplay.forward(request,response);



        }

    }





    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">

    /** Handles the HTTP <code>GET</code> method.

     * @param request servlet request

     * @param response servlet response

     */

    protected void doGet(HttpServletRequest request, HttpServletResponse response)

    throws ServletException, IOException {

        processRequest(request, response);

    }


    /** Handles the HTTP <code>POST</code> method.

     * @param request servlet request

     * @param response servlet response

     */

    protected void doPost(HttpServletRequest request, HttpServletResponse response)

    throws ServletException, IOException {
        processRequest(request, response);
    }


    /** Returns a short description of the servlet.
     */
    public String getServletInfo() {
        return "Short description";
    }
    // </editor-fold>
}
```

Appendix 10

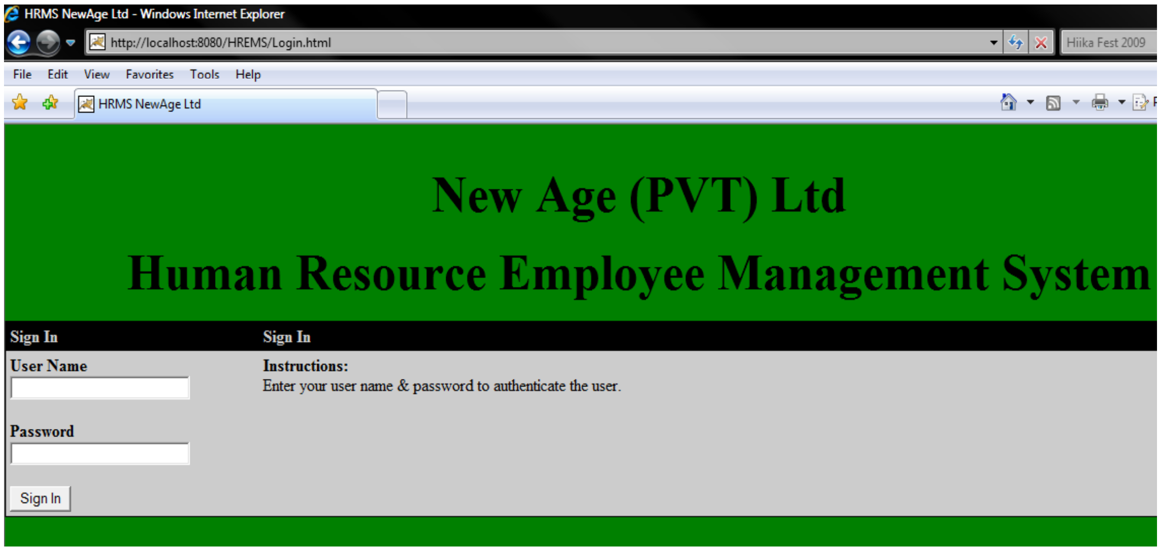
```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>New Age Employee Home</title>
        <style type="text/css">
            body  {background:#EEEED0; color:#663300}
            input {background:#EEEED0; color:#663300}
        </style>
    </head>


    <body>
        <%-- Includes the jspf which provides the standard Music Store page header.--%>
        <jsp:include page = "../WEB-INF/hremsheader.jspf" />

        <h2>Employee Home  <%=session.getAttribute("empId")%> </h2>

        <!--
        This is the custom login page.
        You must use these action and form field names
        for a custom login page. j_security_check is the container login method
        -->
        <form name="details" method=post action="/ems-war/ErrorController">

            <table>
                <tr>
                    <td><a href="/ems-war/EmployeeController"> View personal details</a></td>
                </tr>
                <tr>
                    <td><h4>If errors are found please type it below & report to administration.</h4></td>
                </tr>
                <tr>
                    <td>
                        <textarea name="error" rows="10" cols="100">
            </textarea>
          </td>
        </tr>


      </table>
      <input type="submit" value="SendError" name="Send">
      <br><br>
      <a href="../common/LoginErrorPage.jsp">LogOut</a>
    </form>



  </body>
</html>
```

Appendix 11


EmployeeDataEnterForm - Windows Internet Explorer
EmployeeDataEnterForm
Page
Tools
Submit
Computer | Protected Mode: Off
100%

New Age Corporations >>HRManagerHome>>New Employee Data Recording

User Name*	
Password*	
Name	First Last
Email	
Gender	○ Male ○ Female
Birth Date	MM DD YYYY
Department	
Address	
Qualification	
Skills	
Current Project	
Placement Country	
Placement Company	
Training Period	
Annual Income	
Loans	


HRMS NewAge Ltd - Windows Internet Explorer
http://localhost:8080/HREMS/Login.html
Hiika Fest 2009
File Edit View Favorites Tools Help
HRMS NewAge Ltd

New Age (PVT) Ltd

Human Resource Employee Management System

Sign In

User Name

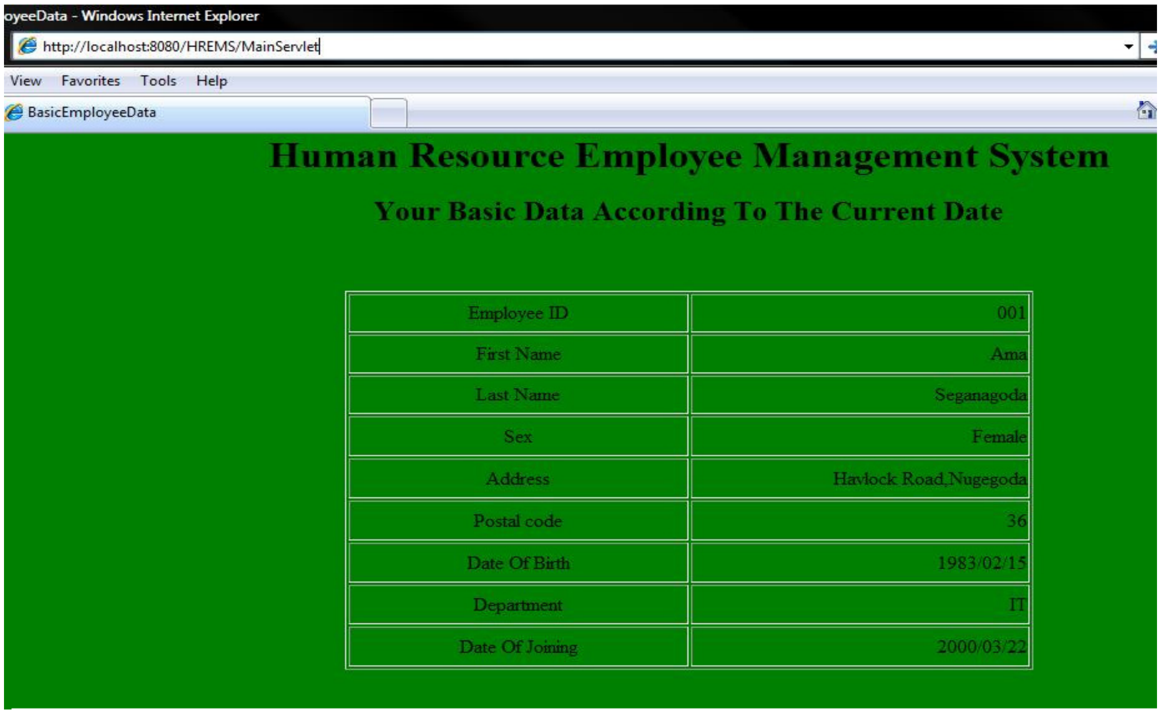
Password

Sign In

Sign In

Instructions:
Enter your user name & password to authenticate the user.

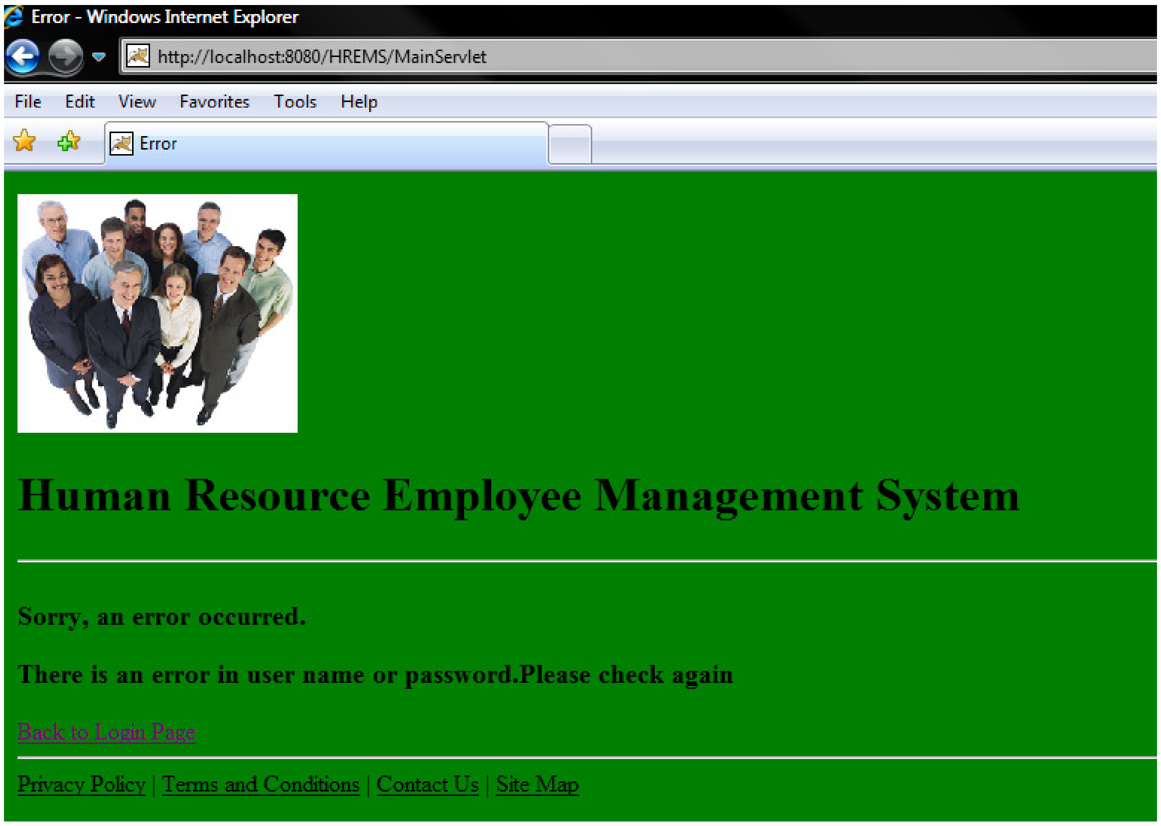
-


oyeeData - Windows Internet Explorer
http://localhost:8080/HREMS/MainServlet
View Favorites Tools Help
BasicEmployeeData

Human Resource Employee Management System

Your Basic Data According To The Current Date

Employee ID	001
First Name	Ama
Last Name	Seganagoda
Sex	Female
Address	Havlock Road,Nugegoda
Postal code	36
Date Of Birth	1983/02/15
Department	IT
Date Of Joining	2000/03/22


Error - Windows Internet Explorer
http://localhost:8080/HREMS/MainServlet
File Edit View Favorites Tools Help
Error

Human Resource Employee Management System

Sorry, an error occurred.

There is an error in user name or password.Please check again

Back to Login Page

Privacy Policy | Terms and Conditions | Contact Us | Site Map