

Distributed Resume Management System Software Architecture Document

Version 1.2

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

Revision History

Date	Version	Description	Author
25/November/2001	1.0	Software Architecture Document generated using Visio.	Chandra Shekara
20/November/2001	1.1	Business Model and Process Elaboration	Naveed Ahmad
02/December/2001	1.2	Collaboration Diagrams	Naveed Ahmad

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

Table of Contents

1. Brief Description	5
2. References	5
3. Architectural Representation	5
4. Architectural Goals and Constraints	5
5. Use-Case View	6
5.1 Architecturally-Significant Use Cases	6
5.1.1 Create Employer Account	6
5.1.2 Search General Resume Pool	7
5.1.3 View Job Postings	7
5.1.4 Create Job Posting	7
5.1.5 Delete Job Postings	7
6. Data Model View	8
7. Logical View	8
7.1 Architecture Overview – Package and Subsystem Layering	9
7.1.1 User Interface	9
7.1.2 Controller (User Task Services)	9
7.1.3 Model (Data Objects Services)	9
7.1.4 Interface to Billing (Domain Service Layer)	10
7.1.5 Web Server	10
7.2 JobApplicant	11
7.2.5 Resume	12
8. Process View	12
8.1 Controller	12
8.1.1 Employer Registration Process	12
8.1.2 Employer Login Process	14
8.1.3 Viewing Job Applicants Process	14
8.1.4 View Resume Process	14
8.1.5 Delete Resume Process	14
8.1.6 View Job Postings Process	14
8.1.7 Add Job Postings Process	14
8.1.8 Delete Job Posting	15
8.1.9 View Job Posting Process	15
8.1.10 Update Job Posting Process	15
8.1.11 Candidate Search	15
8.1.12 Job Applicant Register Process	15
8.1.13 Job Applicant Login Process	15
8.1.14 Job Applicant View Resume List Process	15
8.1.15 Job Applicant View Resume Process	16
8.1.16 Job Applicant Update Resume Process	16

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

8.1.17 Search Job Postings Process general	16
8.1.18 Post Resume to Jobs	16
10 Deployment View	20
10.2 Distrimon Web Server	21
10.3 Tomcat Web Server	21
10.4 Application Server	21
10.5 Data Server	21
11 Size and Performance	21
12 Quality	21

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

1. Brief Description

This Software Architecture Document provides an architectural overview of the Distributed Resume Management System (Distrimon). The Distrimon system is being developed by Distrimon Project team to support online management of resumes and job postings.

2. References

Applicable references are:

1. Distrimon Vision Document
2. Distrimon Project Plan
3. Use Case Specification
4. ER Diagram
5. Class Diagram

3. Architectural Representation

This document presents the architecture as a series of views: use case view, process view, logical view, data view deployment view, and implementation view. These views are presented using the Unified Modeling Language (UML).

4. Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

1. Distrimon System must be interfaced with an existing vendor Electronic Billing System to support billing of an Employer subscription to Distrimon service.
2. All job applicant and employer functionality must be available from both LAN PCs and remote PCs with Internet dial up connections.
3. The Distrimon System must ensure complete protection of data from unauthorized access. All remote accesses are subject to user identification and password control.
4. The Distrimon System will be implemented as a thin-client system-using Network computing Architecture.
5. All performance and loading requirements, as stipulated in the Vision Document must be taken into consideration as the architecture is being developed.
6. The application will be developed using the model, view, and controller architecture.
7. Most probable development platform would be JSP (view), servlets (controller) and JavaBeans(model).

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

5. Use-Case View

The Use Case View is important input to the selection of the set of scenarios and/or use cases that are the focus of iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

The Distrimon use cases are:

- Create Employer account
- Search General Resume pool
- Login Partner Employer
- Search Company Resume Pool
- View Job Postings
- Create Job Posting
- Delete Job Posting
- View Resume for a job posting
- Create Applicant Account
- Login Job Applicant
- View Personal Resume
- Create Resume
- Update/Delete Resume
- Search Job Postings at Partner Employer Website
- Search Job Postings at Central Website
- Submit Resume for a Job Postings

The Job Applicant, Partner Employer or Non-partner Employer initiates these use cases. In addition, interaction with the external Electronic Billing System occurs. Please see the Use Case Specification for details of the use cases.

5.1 Architecturally-Significant Use Cases

5.1.1 Create Employer Account

Brief Description: This use case allows an Applicant or an Employer to register to use the Distrimon services. The main actors of this use case are the Non-partner Employer who become Partner Employers after registration. The Billing System is an actor involved within the create employer account use case.

5.1.2 Search General Resume Pool

Brief Description: This use case allows employers to search the resume repository for resumes of applicants they might be interested in hiring from the general resume pool.

5.1.3 Login - Partner Employer

Brief Description: This use case authenticates a Partner Employer before allowing them into their restricted area of Distrimon.

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

5.1.4 Search Company Resume Pool

Brief Description: This use case allows Partner Employers to search the resumes that have been submitted in response to their job postings.

5.1.5 View Job Postings

Brief Description: This allows for the employer to look as all the job postings.

5.1.6 Create Job Posting

Brief Description: This use case allows users to add new job postings along with the job details and description,

5.1.7 Delete Job Postings

Brief Description: This allows the partner employer to delete a job posting.

5.1.8 View Resumes for Job Posting

Brief Description: This allows the employer to view the job applicant's list and their resume to a job posting.

5.1.9 Create Applicant Account

Brief Description: This allows for registering of a new job applicant into the distrimon system.

5.1.10 Login Job Applicant

Brief Description: This allows a job applicant to login to the system to update his resumes or information.

5.1.11 View Personal Resumes

Brief Description: This use case allows the job applicant to view his resumes list and each resume.

5.1.12 Create Resume

Brief Description: This allows the user to create new resumes while he is logged into the system.

5.1.13 Update /Delete Resume

Brief Description: Allows a job applicant to update a previously posted resume.

5.1.14 Search Job Postings at Partner Employer Site

Brief Description: This use case allows a job applicant visiting a partner employer to search for the job postings there.

5.1.15 Search Job Postings at Central Website

Brief Description: This use case allows for job applicants to search jobs with public access on a central web site.

5.1.16 Submit Resumes for a Job Posting

Brief Description: This allows for the job applicant to post his resume against a job posting.

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

6. Data Model View

The data model gives the way that the entities are related to each other. The database schema for the Distrimon application is based on this diagram. It forms the core upon which the business class layer is modeled.

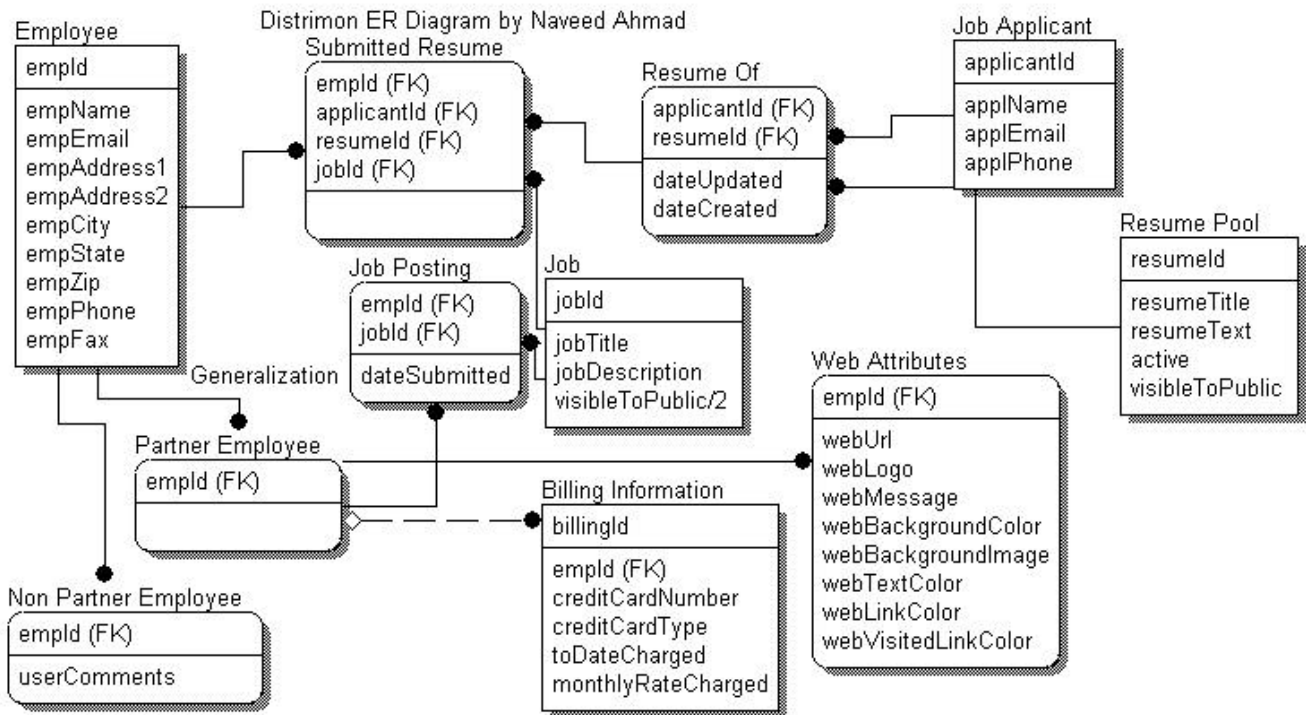


Fig 3. ER Diagram (Database View)

The entity relation ship diagram shows how the entities are related to each other. It also shows the detailed attributes of each entity. The ER Diagram describes the entities in detail. These will also be reflected in the Model or the business logic diagrams.

7. Logical View

The logical view of the Distrimon system is comprised of the 7 main packages: User Interface, Model, Controller, Database, WebServer, Searches and Billing Interface

The User Interface Package contains Active Server Pages (ASP or JSP's) for each of the web page that the actors use to communicate with the System. Controller (User Task Package) contains control classes, which coordinate all activities between ASP or JSP and backend services. This includes login support, manage user account, manage applicant resume, and manage job posting.

The Billing Interface (Domain Services Package) contains classes for interfacing with the finance system.

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

The Data Service Package (Model) contain classes to handle database activities, which includes Create, Retrieve, Update, and Delete of a user account, resume, and job posting. These are the core business classes wrapping over the database tables.

7.1 Architecture Overview – Package and Subsystem Layering

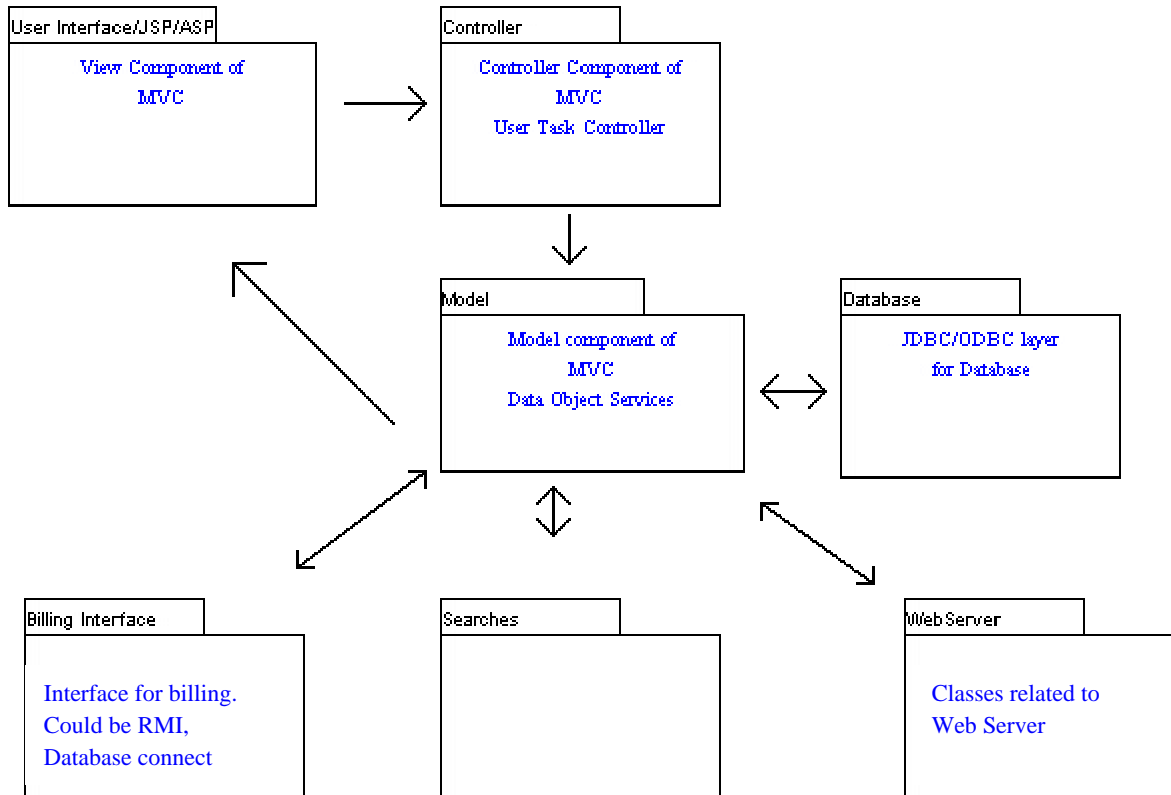


Fig 4. Package Diagram

7.1.1 User Interface

This user interface layer has all the screens that the user sees. This layer depends upon the User Task Services or the Controller; that straddles the separation of the client from mid-tier services.

7.1.2 Controller (User Task Services)

The User Task Services layer has all the controller classes that represent the use case managers that drive the application behavior. This layer represents the client-to-mid-tier border. The User Task Services layer depends upon the Data Objects layer (Model).

7.1.3 Model (Data Objects Services)

The Data Objects layer has all the entity classes that represent the "things" in the domain of the application. These reside on the server and use the web server and database interface service classes to assist in their responsibilities.

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

7.1.4 Interface to Billing (Domain Service Layer)

The Domain service layer provides a buffer between Distrimon system and Electronic Billing System.

7.1.5 Web Server

The web server classes provide the framework for dynamic web content generation. Provides classes for form values retrieval and session management. These classes are usually provided by the web servers api's.

7.2 Model , User Data Services

The following is the core business classes wrapping the core entities and their relations of this application. This wraps over the Entity Relation Diagram from which the database schema was generated. Each of the class which also represents an entity or relation ship in the database implements the DBOperation interface. The database schema relationship and business class model relationship go hand in hand. Therefore each class is responsible to implement the operations interfacing the class to the respective table in the database. i.e. addRecord, deleteRecord, updateRecord, populateRecord. For example if we want to populate a Employee class with a particular record a class user will populate the key for this class i.e. empId and implement the method populateRecord() such that all the other attributes of the class are retrieved from the table Employee. If the key were not populated the method would throw an SQL exception.

Here is a description of each of the model classes and how they collaborate with each other.

7.2.1 Employer

Employer class represents the general entity employer the one responsible for looking and reviewing resumes and posting jobs. This class has the responsibility of interfacing directly to the Employer table in the database. Also it provides means to set and get attributes related to the Employer (general). By implementing the interface DBOperation one may retrieve, delete, update or add a new record into the Employer table. It also provides important methods for the controller classes (which we will visit in the process view) and which expose its relationships with other classes like Jobs, Job Applicants and Resume.

7.2.2 Partner Employer

This class is one of the most important business classes in the package. It inherits the Employer class and the capabilities i.e. interfacing with the Employer table. It also provides the way to retrieve information from the Partner Employer table (which also inherits the Employer table in the database schema). For example given a partner employer class one can retrieve all the job applicants for the employer by calling the method Vector getAllApplicants(). This class can be used by a controller class to retrieve the vector of Applicants and pass it to the view i.e. JSP or ASP. The JSP can simply display all the job applicants with the desired layout. Like the other classes in this package the class implements the DBOperation methods and it can use the super classes implementation as part of its implementation. For example if a new partner employer is to be added into the database the addRecord will first call the super.addRecord so a record is added into the Employer table first. Then it will implement it to add a new row in the inherited table Partner Employer(with the foreign key from Employer table).

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

7.2.5 Resume

This class is the wrapper for the resume table. This allows for addition, deletion, updation, of the resumes. Given a resume object you may retrieve the JobApplicant.

7.2.6 AuthenticateUser

This class is the starting point for an application. This class allows authenticating a user given the userid and password. This not only tells if the user is an Employer or a JobApplicant but also instantiates a class object and returns to the user so he may further navigate other relations from these core classes.

8. Process View

The Process Model illustrates Specification level of the Resume Management classes organized as executable processes. Processes exist to support Login, user account functions, resume functions, job posting functions, and access to the external Billing System. The Controller class diagram (Fig. 7) shows the various Controller classes that are involved the processes diagram below.

8.1 Controller

This package consists of all the controller classes, which will be used by the web forms to post information to. These classes will then instantiate the model classes, set and get attributes, invoked operations and forward to JSP files the model classes (using session objects). The view can display the information from the model classes in the desired layout. Here is the description of all the processes and how the Controller classes come into play as the mediators between the view JSP's and the model business classes. In the Java World the controller classes will be subclasses of the Servlets, which are traditionally used as the control in the web architecture.

8.1.1 Employer Registration Process

In this process a set of forms, controller manager and model class is involved.

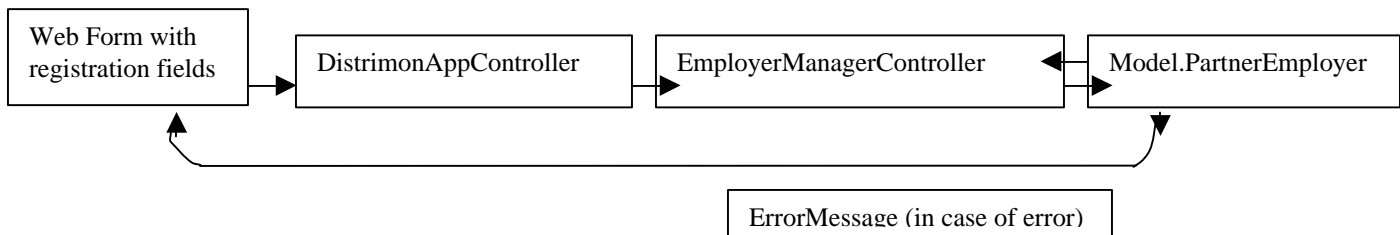


Fig 6. Flow for registration

The above outlines the process of the registration process, where the view, controller and model are in use. The user fills in the web forms, he submits to a controller. The main controller class `DistrimonAppController` determines which `ManagerController` is to handle such a request. It does this by looking at the form id (a hidden field in the form), by using the map from form id to the controller class. All these controller classes inherit from a `WebInfrastructure` class, which provides mechanism to retrieve the Request and Response classes, which in turn allow the controller classes to retrieve the form values and put the model classes into the session. The request is forwarded to `EmployerManagerController` class. Depending on if it was a POST or a GET request from the form, it determines which class method to invoke given the formid. In this case the `createAccount` gets called. The `createAccount` then instantiates a

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

8.1.2 Employer Login Process

This process also follows the above diagram. A user fills in his user id and password and the DistrimonAppController forward to the EmployerManagerController, which in turn uses the AuthenticateUser business class to perform the authentication. The login process is the starting process to enter the system for both the partner employer and the registered Job Seeker. If the user is an employer it also instantiates an Employer object and sets it in the session.

8.1.3 Viewing Job Applicants Process

After a employer logs in he selects an option for viewing the job applicants. When he selects this option the DistrimonAppController is invoked which forwards it to the EmployerResumeManagerController. This uses the getJobApplicants. This method retrieves the Employer method session object and invokes its getAllApplicants method retrieve a vector of all JobApplicants who applied to this employer. The getAllApplicants makes the appropriate query on the submitted Resume and Job Applicant tables to populate the JobApplicant objects and create a vector. This can be passed to the view, which can display the list in the desired layout.

8.1.4 View Resume Process

The applicant may select the job applicant from a list. The view will store the applicant id and resume id with the option in the view. Once the user selects the applicant resume option to view, the DistrimonAppController will forward the request to EmployerResumeManagerController. This will invoke the getResume method of the controller. It will retrieve the applicant id and instantiate a Resume object, set its resumeId and call the retrieve row. This object will be then forwarded to a JSP which will show the contents of the resume to the Employer.

8.1.5 Delete Resume Process

The applicant selects delete option while viewing the resume. The resume id is forwarded to the controller and the deleteResume method called. The controller will retrieve the partner employer object from the session. It will call the deleteResume method and pass it the resumeId and JobApplicant id. This method will delete a row from the submitted Resume table. And forward to a JSP which will repeat the View Job Applicant process.

8.1.6 View Job Postings Process

The employer will select the view job posting option from a menu. The DistrimonAppController will forward to the EmployerJobPostingsManagerController which will call the getJobPostings method. This method will retrieve the PartnerEmployer object from session and call its getJobPostings method. The method will query the database and return a vector of objects of Jobs. The controller will forward to a JSP, which will display the list of job postings.

8.1.7 Add Job Postings Process

The employer will fill in a form with the new job postings attributes and submit the form. This will call the DistrimonAppController which will forward to the EmployerJobPostingsManagerController. This will call the addJobPosting. This object will instantiate a JopPosting object and retrieve the PartnerEmployer from session. Will call the method addJobPosting and pass it the object of JobPosting. This will add the new

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

record into the Job Posting and Job table and forward it to the View Job Posting Process.

8.1.8 Delete Job Posting

The same process as add Job Posting Process, but will call the employers deleteJobPosting method and the view passes the jobId as a hidden field in the view. After this the controller passes to View Job Posting Process.

8.1.9 View Job Posting Process

When a user selects a job posting from the view, the controller is passed the job id to the controller. The controller calls the getResume method. This method instantiates a Resume object and sets its resumeId (retrieved from the form hidden value), and calls its populateRecord method, which populates all the fields of the object from the table. The controller forwards to a JSP which displays the job postings in an editable mode.

8.1.10 Update Job Posting Process

The user will change the values of the job postings and press the update button. This will invoke the controller which will again create a Resume Object sets its resume id and the other fields and call its updateRecord method. This will update the record row with new values. Then the controller passes onto the View Job Posting Process.

8.1.11 Candidate Search

The employer may search on strings for job applicants throughout the database with resumes with visibility as public. The controller will pass control to the EmployerResumeManagerController and call its searchResume method. This method will retrieve the search string and instantiate the ResumeSearch Object and sets its searchString and call the search method. It will then call the getResumesPerJobApplicant. This would return a Vector of ResumePerJobApplicant objects. This controller would then pass this vector to a JSP which could then display the JobApplicants along with their resumes.

8.1.12 Job Applicant Register Process

A Job Applicant would fill a form with the registering values and submit it to the DistrimonAppController which will then forward to the JobApplicantManagerController. This will call the createJobApplicant method. Like the createEmployer method it will retrieve the values from the form instantiate a JobApplicant object sets its attributes and call the addRecord method. This will add a new job applicant. In case of an error will populate an ErrorMessage object and forward it to an error jsp.

8.1.13 Job Applicant Login Process

Similar to the Employer Login Process. The JobApplicantManagerController creates the AuthenticateUser object which returns a JobApplicant object and forwards to a jsp which displays the JobApplicants attributes.

8.1.14 Job Applicant View Resume List Process

Similar to the View Job Posting Process. The Controller passes a Vector of Resumes to the jsp. This

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

displays the list of Resumes a user has posted. The view stores the resume id as hidden values.

8.1.15 Job Applicant View Resume Process

Similar to view job posting update process for the Employer. The controller passed back the resume object to the view by using the resume id passed by the view to the controller.

8.1.16 Job Applicant Update Resume Process

Similar to the update job posting process for the Employer. The controller creates Resume object populates the attributes from the form and calls the updateRecord method. Forwards to the view resume list process.

8.1.17 Search Job Postings Process general

Similar to the employer search resume process. In this case the controller instantiates the JobSearch object and sets its search string entered by the user. It then calls its search method which does a database query to get a list of all the Jobs per Employer. It returns a vector of JobPerEmployer and the view can display all the matching jobs along with the employers who posted them. In case of the specific employer site only the job postings of that particular employer is shown.

8.1.18 Post Resume to Jobs

A candidate may post a resume to any job posting. The view would send the employer id, candidate id and resume id to the controller. That will instantiate the Partner Employer object and call its add newJobApplication by sending it the candidate id, resume id. This method will add a new row to the Submitted Resume table.

9 Collaboration Diagrams

- 9.1.1 The collaboration diagram describes how the classes of the controller, model and other packages interact with each other. Below are some of the important collaboration diagrams for each of the processes described in the process section.

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

9.1.2 Employer Registration Collaboration Diagram

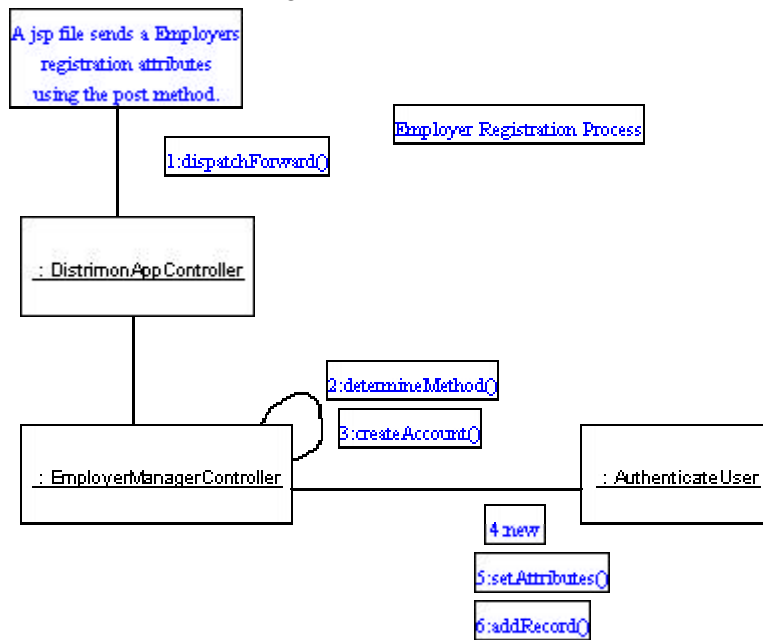


Fig 8. Employer Registration Collaboration Diagram

The above diagram describes how the classes interact with each other by passing messages to register a new employee.

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

9.1.3 Employer Login Collaboration

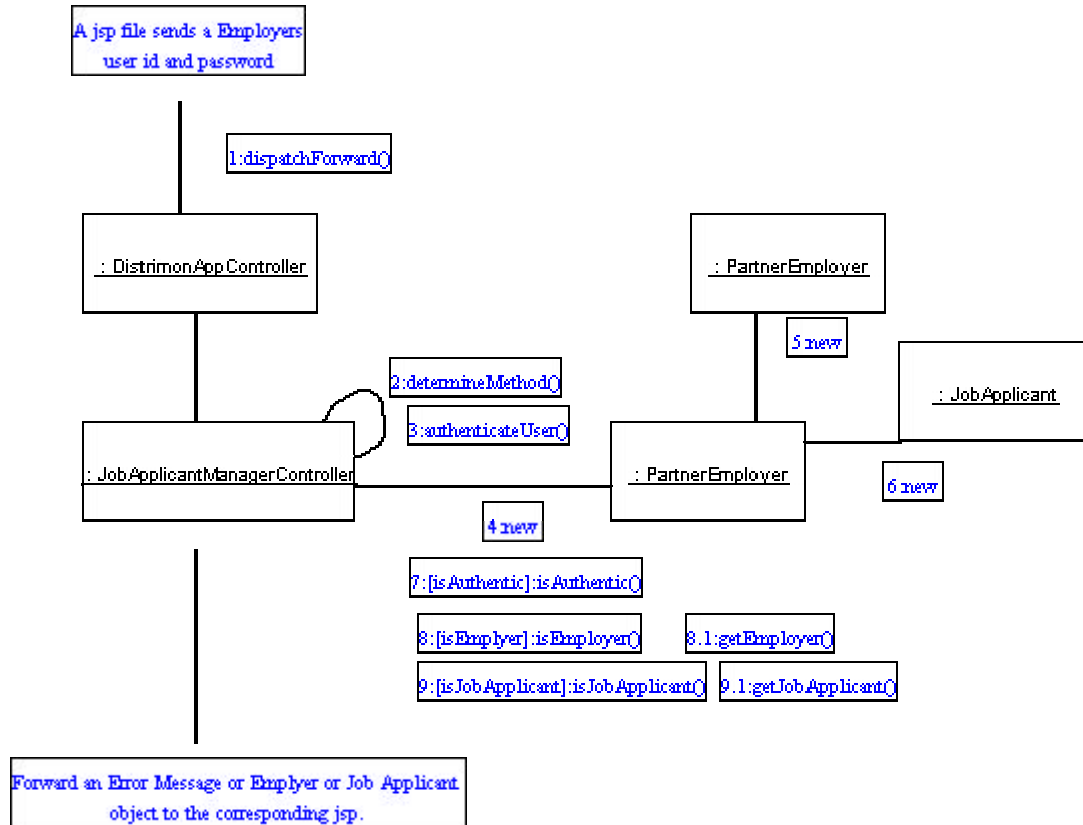


Fig 9. Employer Login Collaboration Diagram

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

9.1.4 Add Job Posting Collaboration

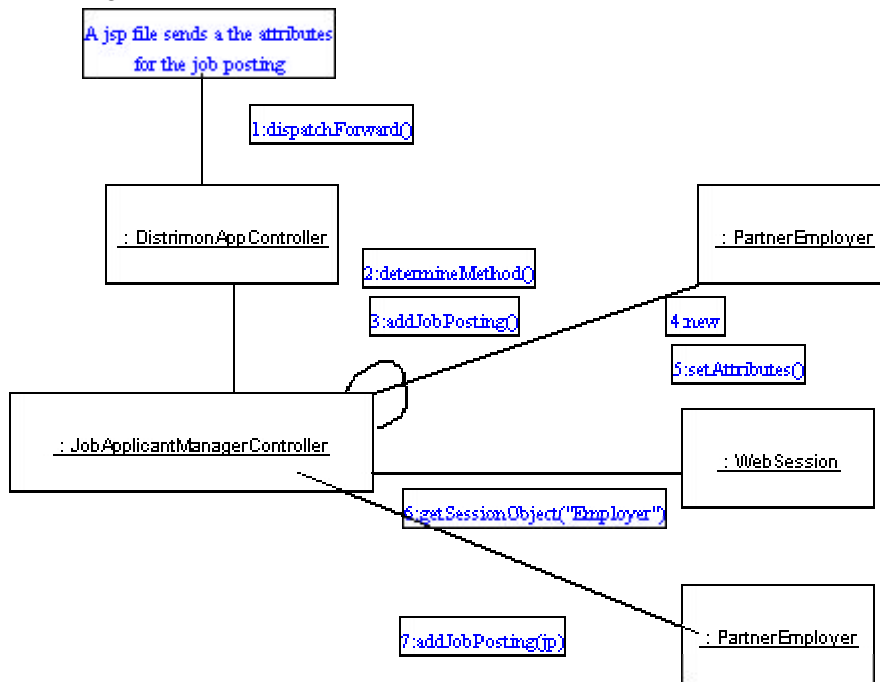


Fig 10. Add Job Posting Collaboration Diagram

9.1.5 Post Resume Collaboration

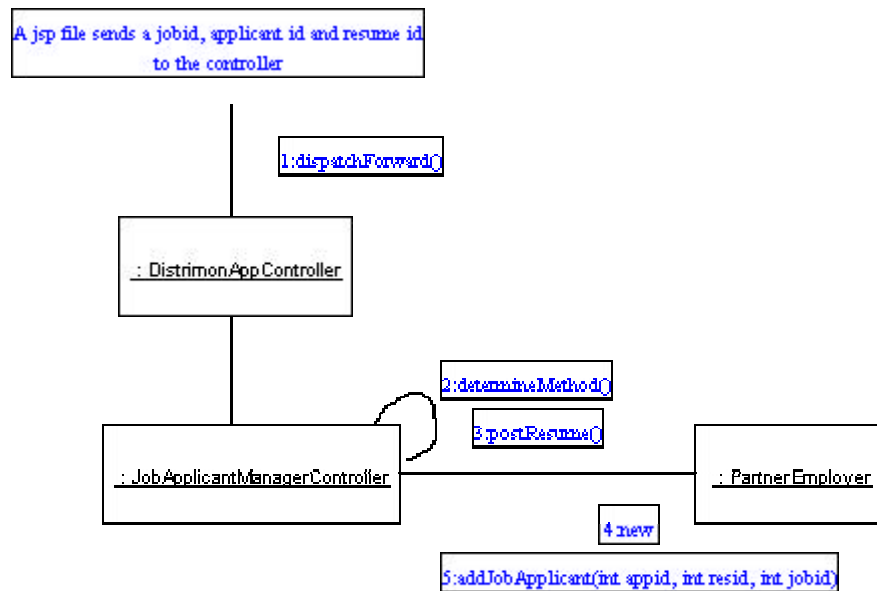


Fig 11. Post Resume Collaboration Diagram

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

10 Deployment View

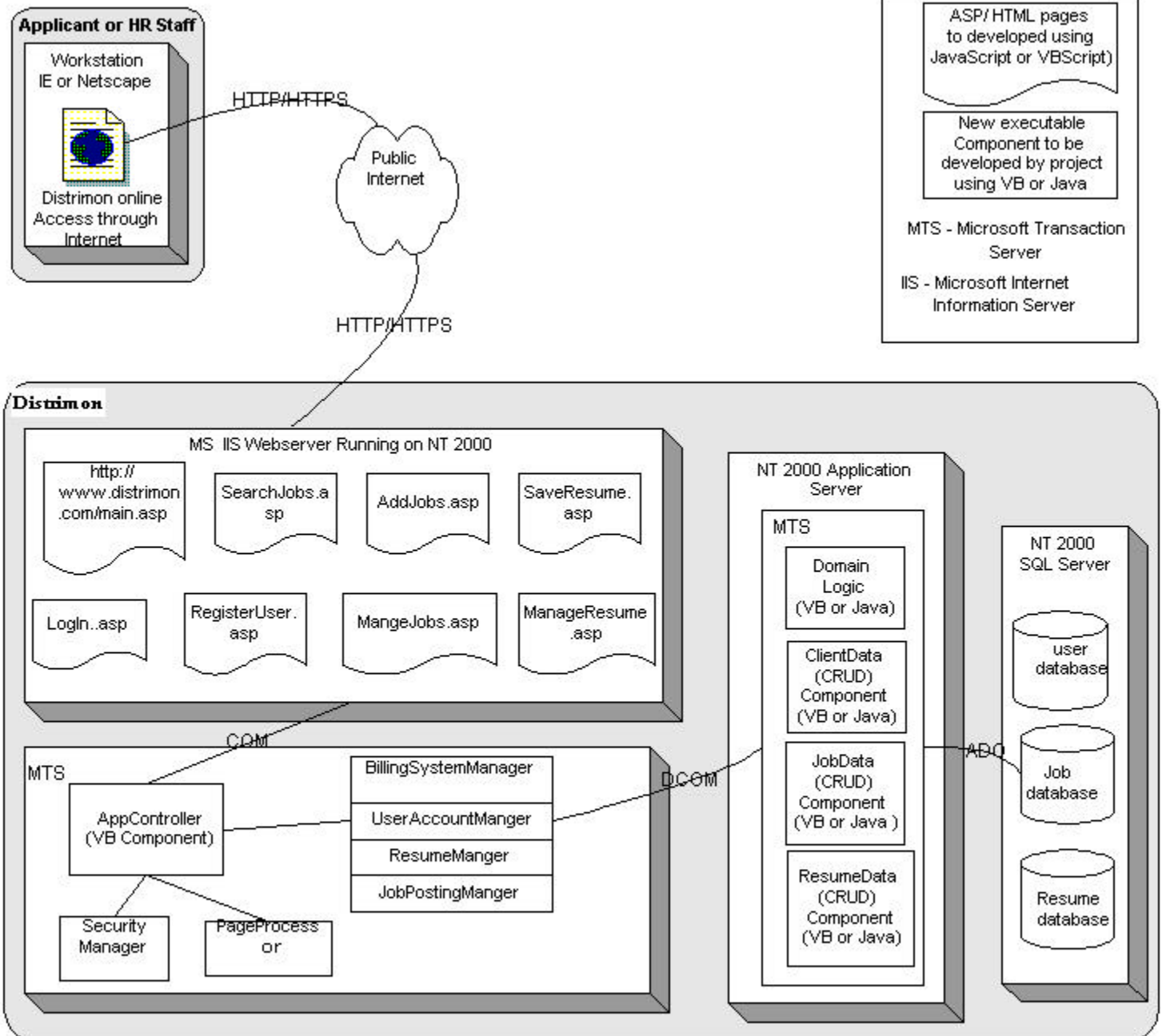
Legend:

ASP/HTML pages
to developed using
JavaScript or VBScript)

New executable
Component to be
developed by project
using VB or Java

MTS - Microsoft Transaction
Server

IIS - Microsoft Internet
Information Server



Distrimon Distributed Component Architecture Using Microsoft Technology
Version : Draft

Distributed Resume Management System (Distrimon)	Version: 1.2
Software Architecture Document	Issue Date: 8/November/2001

10.1 Client PC

Users are connected to the Distrimon web site via public Internet.

10.2 Distrimon Web Server

Microsoft IIS Web server running on Windows 2000 handles users requests.

10.3 Tomcat Web Server

The Java and Servlet container will provide the web services support and run time environment for all the components.

10.4 Application Server

MTS or Tomcat Server Java Container running on Windows 2000, hosts all the mid-tier components.

10.5 Database Server

SQL or Oracle running on Windows 2000 or Unix will be used as the database.

11 Size and Performance

The chosen software architecture supports the key sizing and timing requirements, as stipulated in the Vision document.

1. The system shall support up to 2000 simultaneous users against the central database at any given time.
2. The system shall search resumes or job postings from the central database no more than a 20 second latency.
3. The System shall support both IE and Netscape browsers.

The selected architecture supports the sizing and timing requirements through the implementation of a thin-client architecture. The systems have been designed to ensure that no client components are needed on the Client workstation.

12 Quality

The software architecture supports the quality requirements, as stipulated in the Vision Document.

1. The Distrimon web site should be targeted for IE5.x or above and Netscape 4.7 browsers.
2. The user interface of the Distrimon System shall be designed for ease-of-use and shall be appropriate for a computer-literate user community with no additional training on the System.
3. Each feature of the Distrimon System shall have built-in online help for the user. Online Help shall include step-by-step instructions on using the System. Online Help shall include definitions for terms and acronyms.
4. The Distrimon System shall be available 24 hours a day, 7 days a week. There shall be no more than 4% down time.
5. Mean Time Between Failures shall exceed 300 hours.