PseudoSoft

CENG-491

INITIAL DESIGN REPORT

ONLINE CV PROJECT

Assistant: Ali Orkan Bayer

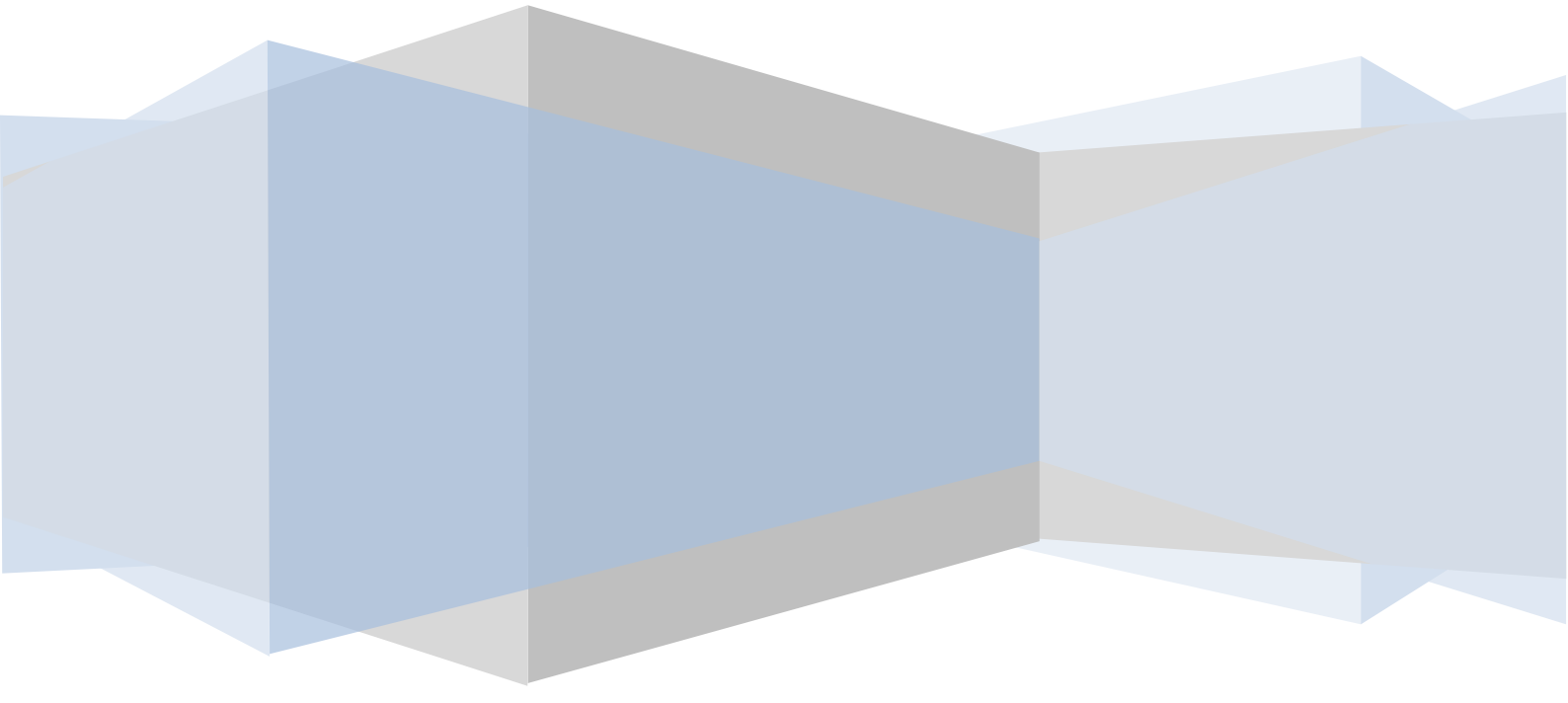


Table of Content

GROUP MEMBERS

Name	Number	E-mail
Gülsüm Selcen Mülazımoğlu	1395276	selcen.mulazimoglu@gmail.com
Ömer Nebil Yaveroğlu	1449248	omernebil@hotmail.com
Mehmet Bahattin Yaşar	1395664	e1395664@ceng.metu.edu.tr
Furkan Kürşat Danışmaz	1394881	k.furkan@gmail.com

A QUICK LOOK TO THE REPORT

PART I – INTRODUCTION

Where "**Problem definition"** introduces you our subject, "**System definition**" introduces you our aim, gives you a quick look to the usage of our system and what we will do, and a general idea of the properties of our system design.

PART II – DATA DESIGN

This part is the most important part of the report for the reader to understand the system and its contents. We begin with defining **data objects** of the job seeking web sites and of our portal, and the **relationships** between them. The **ER – diagrams** will show you how the database of our system and the Job seeking web sites to be simulated look like.

PART III – ARCHITECTURAL DESIGN

Architectural design includes our **design sytle** and **an abstract view** of our **system modules** with the explanations of our decisions. The **state diagrams** will show you the sequential behavior of our system, whereas the **Data Flow Diagrams** will give you an "**abstract look"** to the data, procedures and the control. In other words, DFDs will represent you our system as a whole and the flow of data between the **modules.**

PART IV – USER INTERFACE DESIGN

The user interface part will show you how our portal will look like as a preliminary study.

PART V – PROCESS SPECIFICATIONS

In the process specifications part we gave a **final level of refinement** of the system modules given in the Data Flow Diagrams for giving an abstract view. We tried to explain every service of our design and their contents one by one in detail. **Control Specification** part will give you more detailed look of our system and will show you the **behavior** of our system in different scenarios. "**Sequential Diagrams**" will show the sequential behavior of our system and finally "**Enterprise Service Bus - ESB**" part will mention about integration issues of the services.

PART I - INTRODUCTION

1.1. PROBLEM DEFINITION

In today's world of rapidly expanding technology, we can do many things by means of a computer with an internet connection. For example, we can read newspapers, buy something from a shopping website, or do many banking operations. We can also seek for a job through internet. However, since there are many job seeking web sites, this job seeking process sometimes becomes very boring and time consuming. In order to find a suitable job for ourselves, we must be a member of many job seeking websites, and we must fill a lot of similar forms to describe ourselves.

In companies' point of view, the problem is much more bigger. In addition to loosing a lot of time, the cost of reaching to a suitable employee for the company is another problem to be solved because the job seeking websites require considerable amount of money to publish a job announcement.

In a more general point of view, the number of people who look for a job or an employee is divided among all job seeking web sites. Therefore, the possibility of finding a match between an employee and a job is reduced so much because there are a lot of job seeking websites. If a job seeker wants to see all the suitable jobs for himself/herself, he/she has to sign up to all the job seeking web sites and repeat the same search process from all those web sites one by one. Likewise, if a company wants to reach more job seekers to find the most qualified and suitable one, that company has to leave its job announcements to all of those websites by paying a lot of money to each one.

May be the most dramatical problem occurs when there is a match between a CV and a job which stay in different websites' databases but there is no match in their respective databases. In other words, there may be a job announcement that has no match for a long time in a website, and likewise there may be a CV that has no match for a long time in a different website. That CV and that job may be suitable for each other. However, since a job seeking website doesn’t know anything about the information in other's databases, those

websites can not serve their customers fast and adequately as expected. Hence, they start to lose their prestige and so the customers.

Another problem in today's situation is about management of human resources in the country. İş-Kur which is responsible for the planning of the human resources management in our country requires statistical information about employee discharge and employee acceptance to be able to make a good plan. However, there is no mechanism to provide these required statistical information dynamically to İş-Kur. Therefore, İş-Kur is lack of the most recent and correct statistics about human resources.

1.2. SYSTEM DEFINITION

As a result of the defined problem in “Problem Definition” part of our report, we have defined the main properties of our project, ONLINECV. Although the borders of the project were not clear when it was given to us, after brainstorming with our group members, we made some general decisions about the project. Basically, we will implement an online CV and job announcement distribution web site using web services to solve the problem described in the previous section. Now let me describe how our system will look like, how it will be used and what the requirements are in order to use our system.

First of all, I want to mention the usage of our system shortly. Our system works as a web page similar to Sigortam.net, Akakce.com, etc. Its main difference from other web sites in World Wide Web is that our system uses a new technology, web services, as a mean of communication between websites. By the use of web services, we will reach other job seeking web sites’ services, reach their job announcements, and send CV’s to them from our portal. While doing this, our aim is to use as much simple user interface as possible. When a user wants to send a CV to all these web sites or search for a job, he will only fill the forms in our web site. After the user fills the required forms, we will do all required operations from all other web sites like Kariyer.net, Yenibiriş.com whose web services are available to our portal. If a user becomes a member of our portal, then we make him/her a member of all job seeking sites automatically. If he/she creates a CV in our portal, we distribute that CV to all other websites immediately. Actually we didn’t store any CV or job announcement information in our portal’s database, but we send those information to the other websites to store them. If a

user wants to make a job search operation, he/she gives the job specifications he/she desire to our portal, then we will make a search in all job seeking sites, and compose the information which comes from them and introduce the results in an easy-to-understand way. The user will be able to apply to those results immediately, because sending CV to the job announcements is free of charge.

When we come to the companies' side, we will provide them an easier way to leave their job announcements in a bit cheaper way because we assume that we made a deal with all job seeking websites to make a discount to the job announcements coming from our portal. (We think this is logical, because by means of our portal, those job seeking sites increase their number of customers.) Besides, if a company wants to leave a job announcement, after filling a required form, we will provide a page which contains some price comparisons between job seeking sites, and the company is allowed to select in which web sites it prefers to publish the job announcement. In order to complete this operation, the company must have enough money in our portal's account. (If not, then by means a banking website which uses the money transfer webservice of our portal, they can send money to us. To be able to do this, we added an additional money information in our portal's companies' membership database.) As a result of the selection, we will distribute the job application to the selected web sites, and also make money transfers to those selected web sites.

During those interactions with our portal's users, we will try to make things as simple as possible. In addition to the services which we provide to employees and companies, we will also provide a webservice which returns the most recent statistics about employee acceptance and discharge. This webservice is only available to "İş-Kur". To provide such a service, our portal will use the statictics services of all job seeking websites, then we will collect those statistics and combine them and send to İş-Kur whenever it requests. May be, this service can be provided to some other foundations working for public welfare.

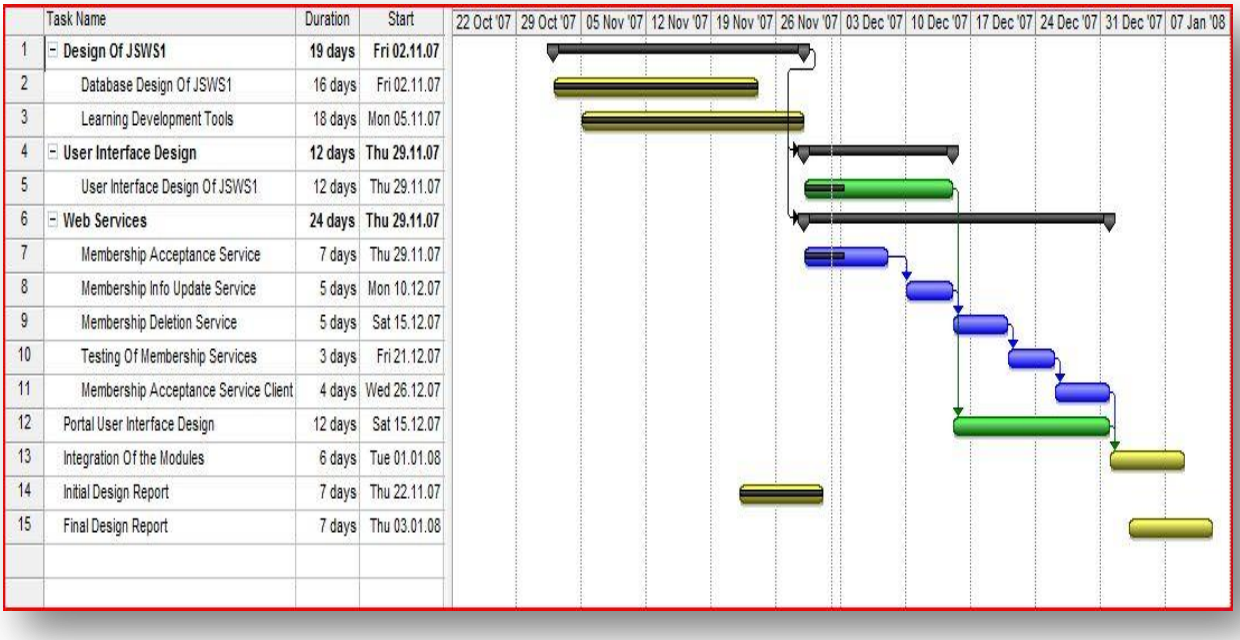
As you can predict from the explanation of our system, our portal requires to use all job seeking websites' web services. However, this is not possible, because those websites does not give such required web services. Therefore, to be able to do our project, firstly we will build two web sites simulating Kariyer.net and Yenibiris.com. After writing the required web services in those simulator web sites , we can start to build our portal's functions.

1.3. WHAT WE HAVE DONE SO FAR

As a group we initially started with the design of the database and the graphical user interface of Job Seeking Web Sites. We completed this period successfully and formed the database design in detail. In the data design part, the database design and the data objects of both the portal and job seeking web sites are explained in detail. As a parallel process we started to learn the development tools. We learned how to write a web service by using "Eclipse" for code development platform and "PostgreSQL" as database management system. On the other hand, to develop user interface we tried several tools one by one, by searching through internet. A responsible person from SoftwareAG explained how we will use the tools given by SoftwareAG like "Application Composer" and we tried to use these tools. However, we had a license key problem while installation and we stated this situation to Efe Akman, the responsible person of our project.

As we are composed of two groups, we are working on two different processes in parallel. While one group is dealing with the user interface design, the other group started to write membership acceptance service.

Now we are in the right place in our time line. We try to meet the deadlines we specified in our time line as possible as we can do.



	Task Name	Duration	Start
1	**Design Of JSWS1**	**19 days**	**Fri 02.11.07**
2	Database Design Of JSWS1	16 days	Fri 02.11.07
3	Learning Development Tools	18 days	Mon 05.11.07
4	**User Interface Design**	**12 days**	**Thu 29.11.07**
5	User Interface Design Of JSWS1	12 days	Thu 29.11.07
6	**Web Services**	**24 days**	**Thu 29.11.07**
7	Membership Acceptance Service	7 days	Thu 29.11.07
8	Membership Info Update Service	5 days	Mon 10.12.07
9	Membership Deletion Service	5 days	Sat 15.12.07
10	Testing Of Membership Services	3 days	Fri 21.12.07
11	Membership Acceptance Service Client	4 days	Wed 26.12.07
12	Portal User Interface Design	12 days	Sat 15.12.07
13	Integration Of the Modules	6 days	Tue 01.01.08
14	Initial Design Report	7 days	Thu 22.11.07
15	Final Design Report	7 days	Thu 03.01.08

Figure 2.1 – Gantt chart of the first semester

PART II - DATA DESIGN & DATA DICTIONARY

2.1. Data Objects of Job Seeking Web Sites – Job Seeker Part

Job Seeker Membership Information: Each job seeking web site need to hold the "username", "password", "email", "secret question", and "answer" for opening sessions to users. Therefore the "Membership information table" will be hold in each of the job seeking web sites.

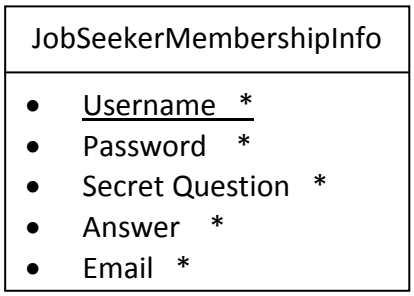


Figure 2.2 –Structure of the "Job seeker membership information" objects

This table has the same structure with the "Job Seeker Membership Information" table in our portal. Note that a person does not have to fill a CV to become a member. Instead, he/she can create CV or CVs after the registration step at any time he/she wants.

When a user wants to become a member of us and job seeking web sites our portal will send all the information about the user to other sites. The membership information of the user such as username and password will be the same in all job seeking web sites with our portal. This provides the user flexibility such that he/she can make operations directly in job seeking web sites without using our portal. However, by using our portal, the user can make changes in membership information held in the all job seeking web sites without signing in to all of them one by one.

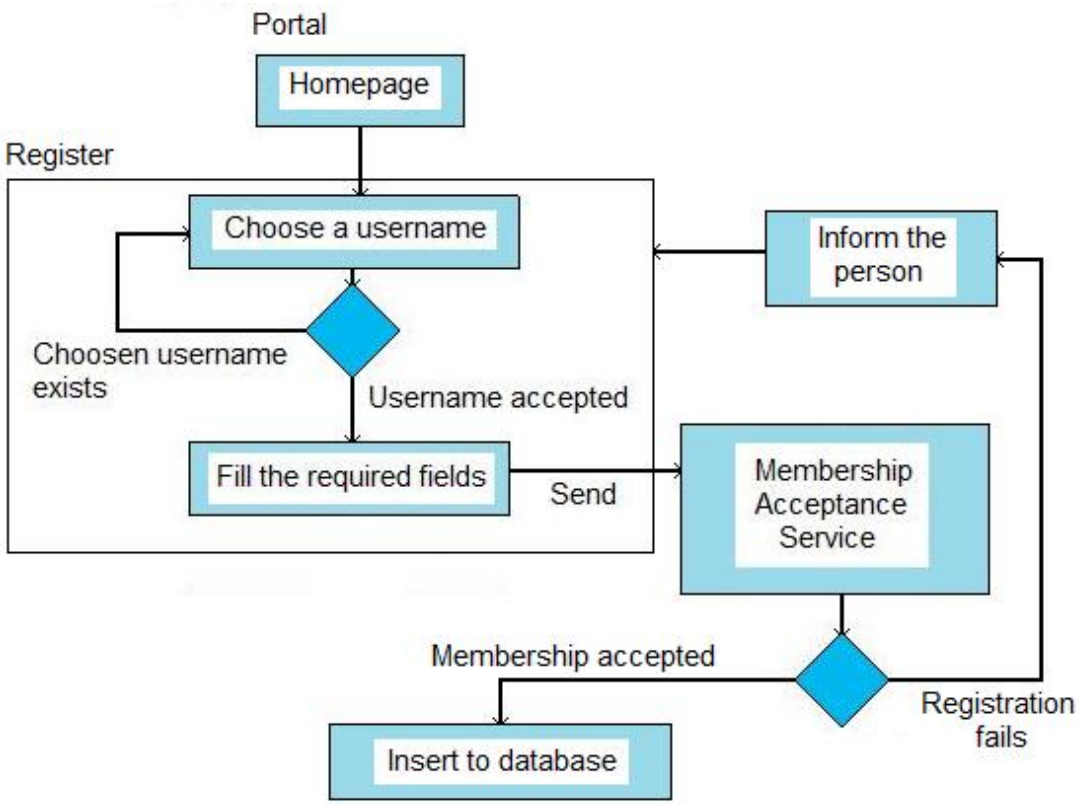


Figure 2.3 Flow chart of the registeration

While registration, since we will give the same username to the user for each job seeking websites, first we need to know that whether the choosen username exists in any of the websites, including our portal. After searching our database, we will send messages to other sites including the username, and “Username acceptance service” running in their server will respond us with a message including a boolean value, true or false, meaning username accepted or rejected respectively. After the person chooses an appropriate username and filling the other required fields, our portal will send the registration information to the other sites, and “Membership acceptance service” running on the servers of the job seeking websites will accept the membership request, save the user to their database, and will return us a message meaning “Membership accepted”. Finally, our portal will insert the user to our database. If somehow registration fails, “Membership acceptance service” will send a message meaning “Registration failed” to our portal and we will inform the person and redirect to the registration page.

CVs : Since a member may want to create more than one CV we decided to create a separate CVs table where each CV is associated with a member with a relationship. A CV object will have the following attributes.

- CVId : For idendifying CVs
- CVName : Uniquely identifies the CV with respect to the CV owner. (Title)
- Current Work Status : Indicating whether the individual works or not.
- Job Experience : Total experiences in year.
- isActive : Whether or not the CV will be seen by companies.
- CvLanguage : The language in which the CV will be presented.

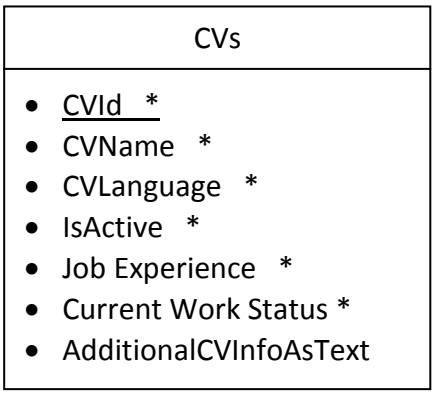


Figure 2.4 – Structure of the "CV" objects

A person may have more than one CV saved. Therefore, instead of embedding all the properties like "Identity Information", "Foreign Languages", "Military Service Information", or "Educational Information" to the CVs table, we decided to hold such informations as separate tables and associate those tables with the CVs. Because separating related fields to a different table, will significantly improve the performance of fetching information from the database.

In the "additional CV information", member can mention about his/her talents not included in the web page.

Data Objects associated with the CVs table

Identity Information: For holdig identity informations of the person like “Name”, “Surname”, “Gender”, “Nationality”, “TC Number” – to be filled by Turkish citizen, and the other indentity informations. An identity information object associated to a CV object will have the following properties:

Figure 2.5 – Structure of the “Identity information” objects

Communication Information: For holding communication information of the individual like “Telephone numbers of the person”, “e-mail”, or “web page”. Each “communication information” object will have the following properties:

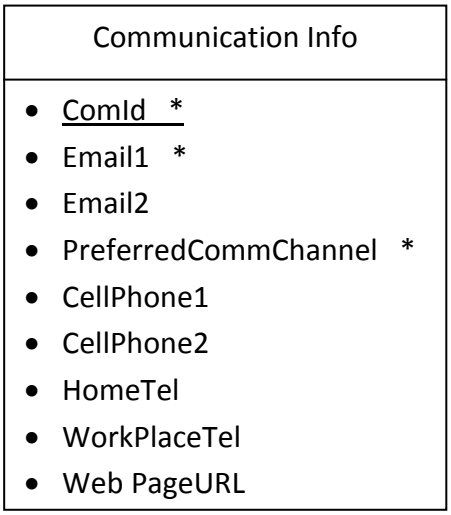


Figure 2.6 – Structure of the “Communication information” objects

"Id" is for identifying the "communication information" object. A member should give at least one e-mail address while filling a CV so that job seeking web sites will have at least one communication way. Other attributes like cell phone numbers or web page URL are not absolutely necessary.

Addresses: For holding address information of the individual. As expected, an address object will include the following properties:

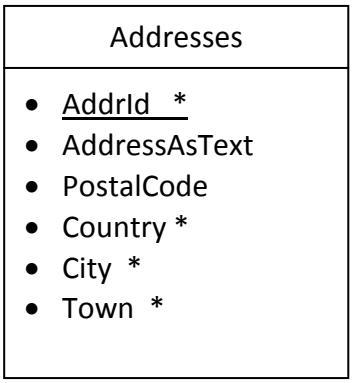


Figure 2.7 – Structure of the "Address" objects

"AddressId" is for identifying the address object. The attributes "Country", "City", and "Town" is required to be filled for each address object to be meaningful. Other than those fields, if the member wants to give more specific address, he/she can fill the "AddressAsText" field.

References: For holding the information about the person referenced by the member of the website. Each reference object will include the "name" and "surname" of the referenced person. Other than the name and surname attributes, "Job status" and "Foundation name" may be filled (Optional).

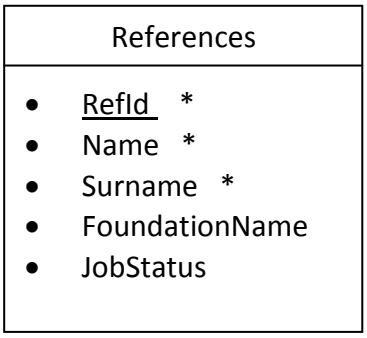


Figure 2.8 – Structure of the "Referenced person information" objects

Military Service: This is required for each member for holding military status of the individual. The female members should choose "Yapıldı/Muaf" from the options.

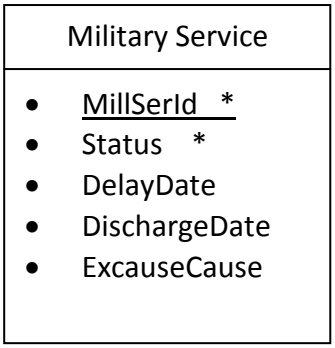


Figure 2.9 – Structure of the "Military Service information" objects

Since we will simulate the existing job seeking web sites like Kariyer.net, we became a member of those sites and we tried to understand their database structure, and we tried to resemble our database design to it. The following screen shot shows how similar our database design for the simulation of job seeking web sites to Kariyer.net.

Figure 2.10 - Kariyer.net > Özgeçmiş Güncelleme > Askerlik Durumu

Driving License Information: Holding driving license information of the members. Each "driving license" object may be associated to one or more CV objects which will be associated to a member. The relationship between the CVs and the Driving License will hold the "issue date".

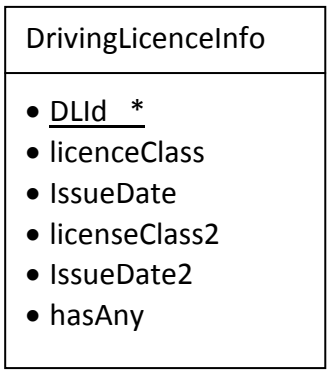


Figure 2.11 – Strcuture of the "Driving License information" objects

The following screen shot shows the “Driving license” part of the Kariyer.net.

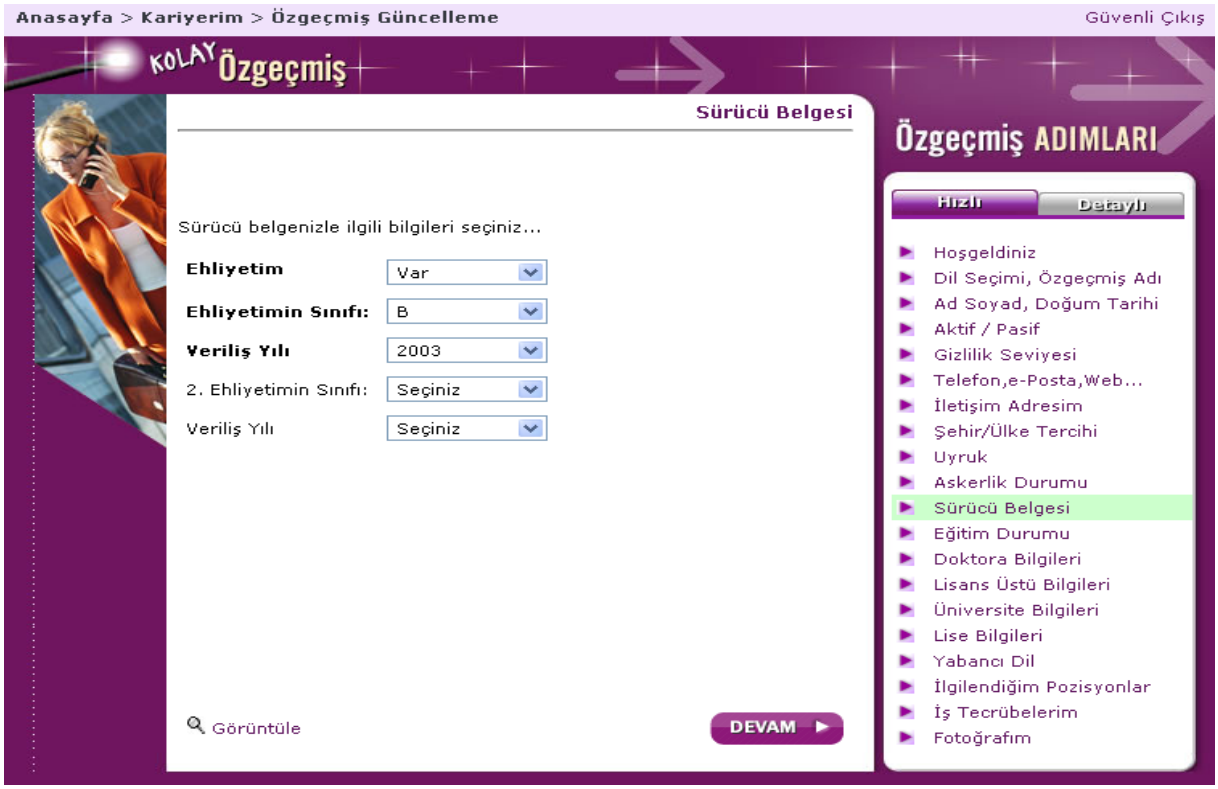


Figure 2.12 – Kariyer.net > Özgeçmiş Güncelleme > Sürücü Belgesi

Preferred Places: A static table holding the locations, namely “city” and “town”. A “preferred place” object will be associated to a CV object with a relationship which enables the database of the job seeking website know the answer of the question “who prefers where to work”.

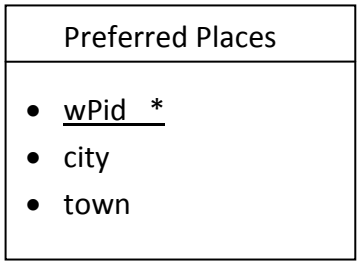


Figure 2.13 – Structure of the “Preferred Places” objects

Here is the preferred places part of the “Yenibiris.com” database. Although the following screen shot is taken from the job announcement creation page, we also include this information related with CVs. Because when we search CVs for unmatched job announcements, we will use this information to be able to match them with CVs.

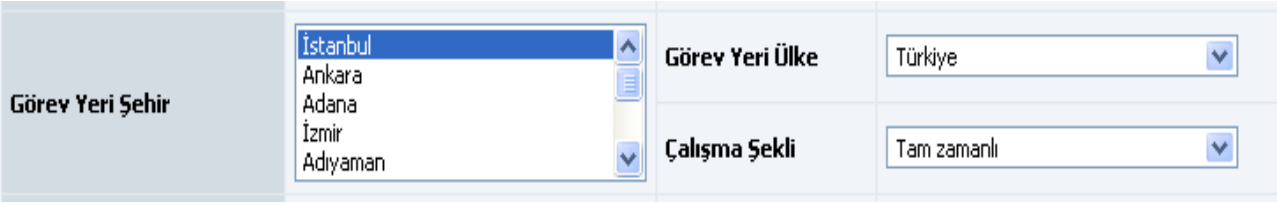


Figure 2.14 – Yenibiris.com > İlan Hazırlama Formu

Preferred Work Status: This table stores static information about work status that a job seeker can accept. The values of this table are "Yarı Zamanlı", "Tam Zamanlı", "Dönemsel", "Sözleşmeli" and "Stajyer". "Create CV" form page will include a combo box, a button and a data table for this information. A job seeker can add the work status to the list in the data table that he/she accepts by selecting an item from combo box and press the button. There will be a relationship between "Preferred Work Status" and "CVs", and the information in the data table is stored into this relationship.

As you can see from the figure above which is taken from the Job announcement creation page, the preferred work status is also included. Again to be able to match CVs and Job announcements easier, we add "preferred work status" to Job seeking web site database related as CV.

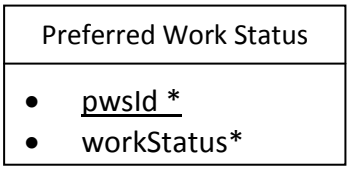


Figure 2.15 – Structure of the "Preferred Work Status" objects

Foreign Languages: "Foreign languages" will be another static table of the job seeking web sites. A foreign language object will be associated with a CV object with a relationship that has "Reading", "Writing", and "Speaking" fields for specifying levels, as shown in the figure below. With this relationship, the database will be able to know "who knows which languages at which level".

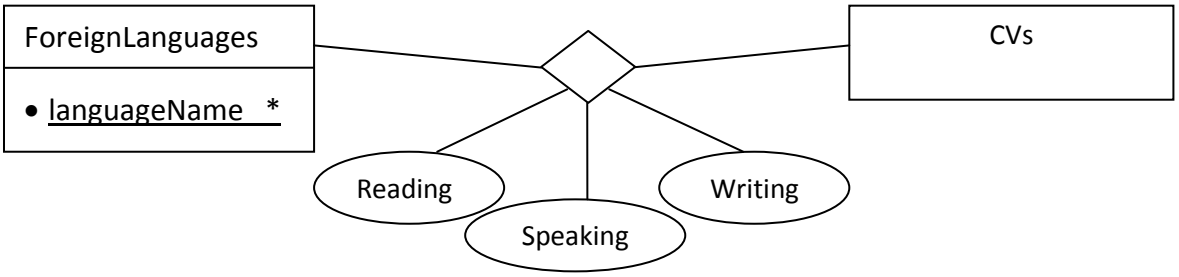


Figure 2.16 – Relationship between "CVs" and "Foreign Languages"

Here is the "Foreign Language" part of the CV creation step in "Kariyer.net.

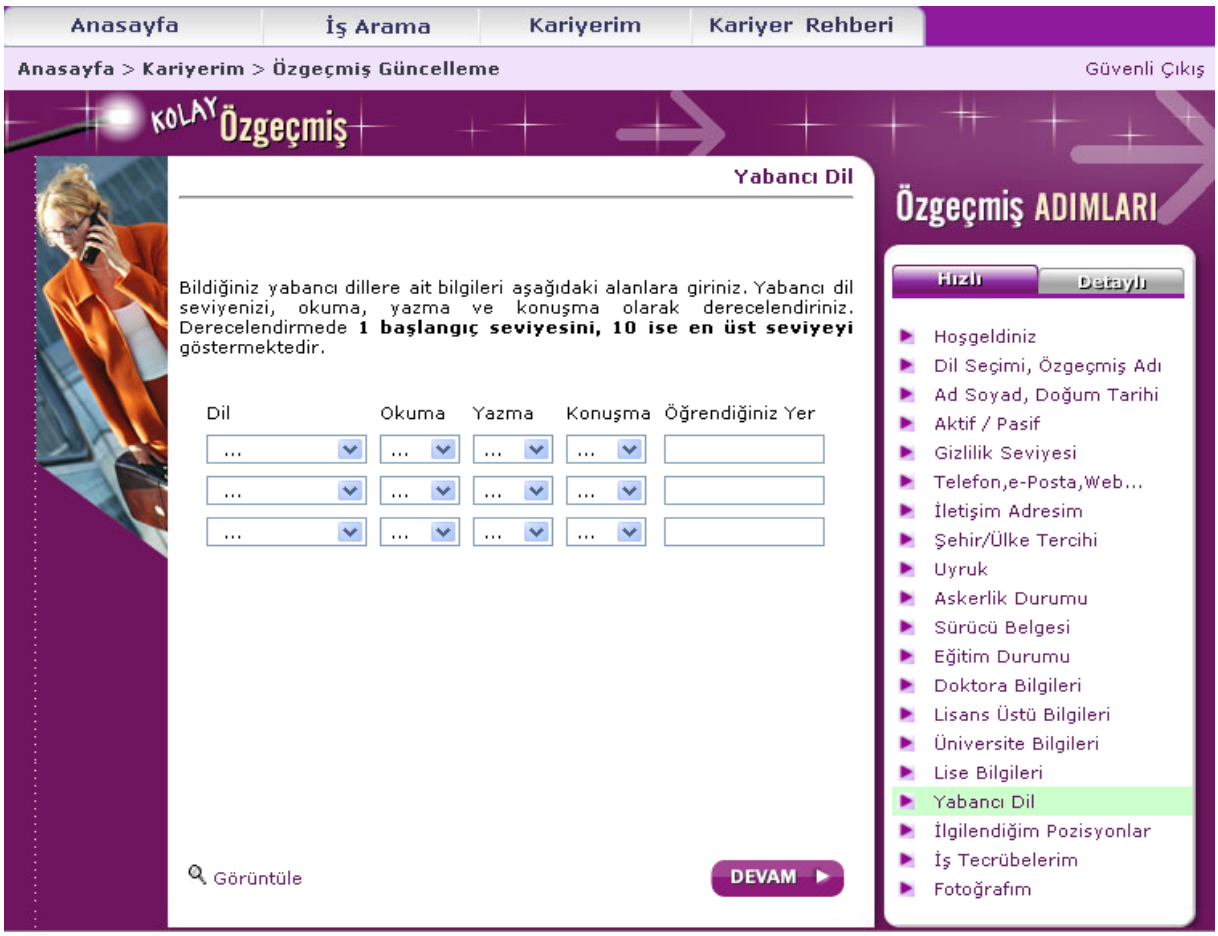


Figure 2.17 - Kariyer.net > Özgeçmiş Güncelleme >Yabanci Dil

Although there is a limit of three of the foreign languages that a member can specify, we prefer to separate this information as an entity so that the member can specify as many languages as he/she likes.

Interested Positions: **"**Interested positions" table is another static table holding "Sector" and "Position" of the job seeking web sites. An interested position object will be associated with a CV object.

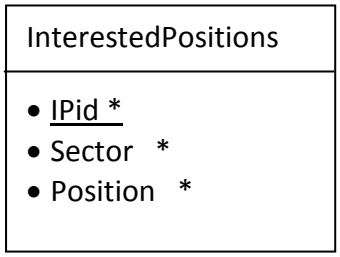


Figure 2.18 – Structure of the "Interested Positions" objects

We inferred "Interested Positions" table structure from the below "İlgilendiğim Pozisyonlar" of the "Kariyer.net".

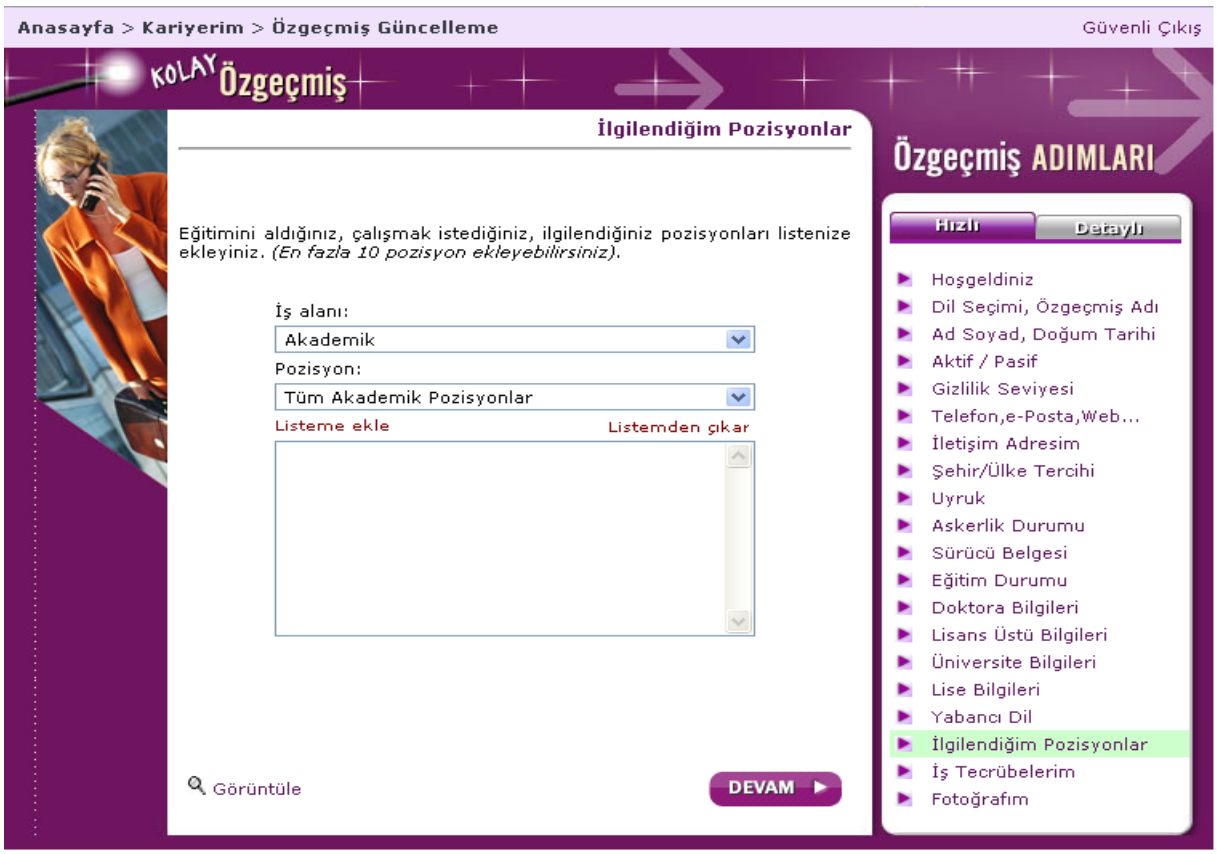


Figure 2.19 – Kariyer.net > Özgeçmiş Güncelleme > İlgilendiğim Pozisyonlar

Experiences: A person may have more than one job experience as he/she may have worked at different places in different positions. An experience object will include the following properties:

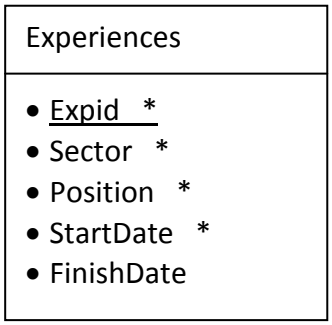


Figure 2.20 – Structure of the "Experiences" objects

Personal Information: Each "personal information" object holding some personal information about the member will be associated with a CV object. A personal information object will be as follows:

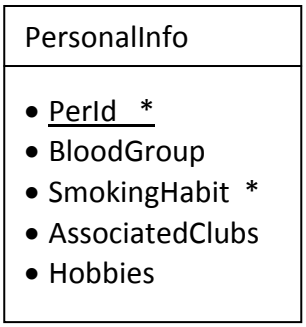


Figure 2.21 –Structure of the "Personal information" objects

Education Information: Each education information object will be associated with a CV object.

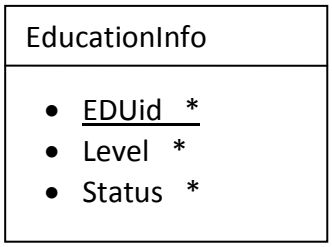


Figure 2.22 – Structure of the "Education information" objects

When we try to match CVs with unmatched job announcements, the "Level" and "Status" fields will correspond to "Minimum educational level requirement" and "Acceptable Education Status" respectively.

To make the meanings more clear, let us give some possible values of these fields. “İlköğretim Okulu”, “Lise”, or “Üniversite” are some of the possible values of “Level” field. On the other hand, “Status” field can take the following values: “Mezun”, “Öğrenci” or “Terk”.

Computer Knowledge: This table stores static information about computer programs which a job seeker can know how to use. Some of the values in this table are “Photoshop”, “Word”, “Excel”, “Powerpoint”, and goes like that. There will be a combo box, a button and a data table in “Create CV” form page. A job seeker can select an item from the combo box, and add it into the list in the data table. The list in that data table actually is stored in a relationship between “Computer Knowledge” and “EducationInfo”.

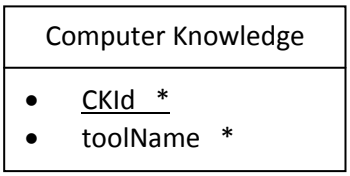


Figure 2.23 –Structure of the “Computer Knowledge” objects

Ph.D. Information: If a job seeker has a Ph.D. degree, he/she will fill the necessary fields in the “Create CV” form, and the information in those fields will be stored to the table below. Although it can be inferred from the names of the fields, let us explain some the fields to make the point more clear.

“GradingSytem” field specifies the university’s grading system. For example, the grading system of METU is 4. “UniversityType” field specifies the type of the university where possible values of that field are “Vakıf”, “Devlet”, and “Yurtdışı”. GradDate and StartDate is for the graduation date and the starting date of the employee to the Ph.D. degree program.

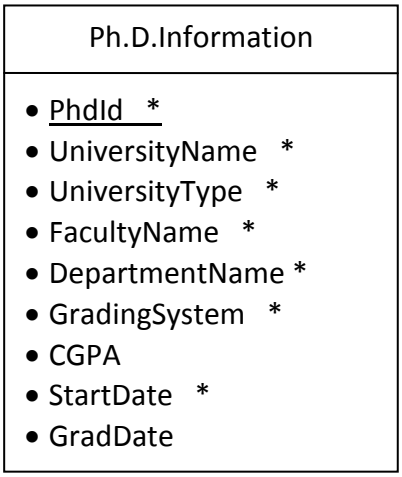


Figure 2.24 – Structure of the "Ph.D. information" objects

Master Information: This table's usage is exactly same with the "Ph.D. Information" table.

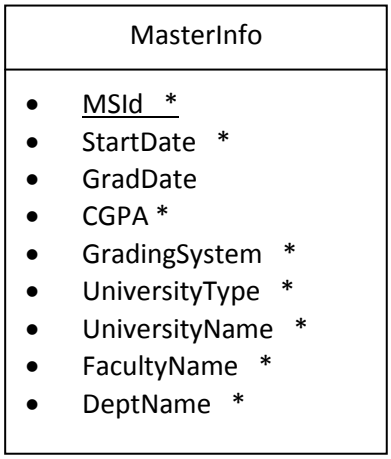


Figure 2.25 – Structure of the "Graduate Information" objects

Undergraduate Information: If a job seeker is a student in a university or is a graduate, he/she will fill the necessary fields in the "Create CV" form. We create the table below to store the information in those fields. Although many of the fields are similar to the fields of "MasterInformation" table, let us explain the different ones. Since some universities can provide education in different time schedules and places, we needed a field to specify this point, and named that field as "EducationType". The possible values of that field are "Örgün Öğretim", "İkinci Öğretim", "Açık Öğretim" and "Uzaktan Öğretim". Likewise, universities' education language can be different. For example, education language of METU is English, whereas some universities' may be Turkish, or French in Turkey. "EducationLanguage" field is for this point. "isDoubleMajor" field will be true if a job seeker is/was registered to a double

major program in his/her university. In that case, we will store those informations in these fields: “FacultyName2”, “DeptName2”, “StartDate2”, and “CGPA2”. Since a person can register a double major program only in his/her university, it is unnecessary to store the university information again, so we didn’t do like that.

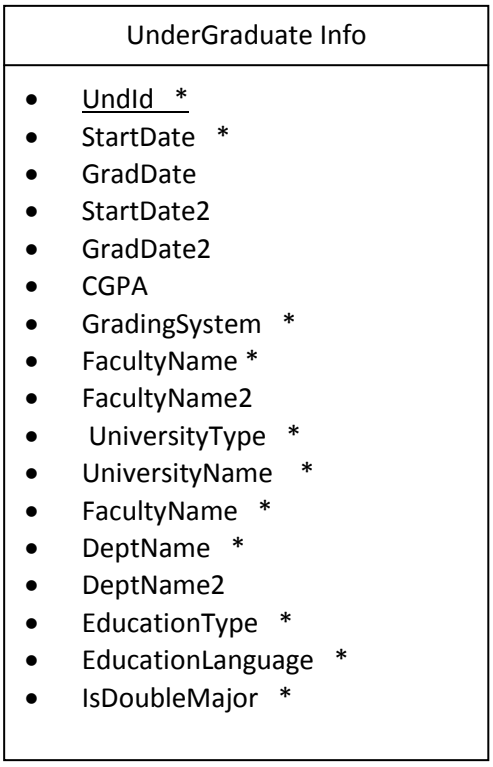


Figure 2.26 – Structure of the “Undergraduate information” objects

High School Information: When a job seeker fills the related fields with high school education in the “Create CV” form, we will store the information into the “HighSchoolInfo” table. In this table, in order to store the type of the high school there is a “SchoolType“ field a few of whose possible values are “Normal Lise”, “Açık öğretim Lisesi”, “Anadolu Lisesi”, “Fen Lisesi”, and goes like that. (There are about 40 high school types.) Branch field specifies at which branch of the high school a job seker is. A few of possible values of that field are “Acil Tıp Teknisyenliği”, “Elektrik”, “Gemi İnşaa”, “Fen Bilimleri”, ”Sosyal Bilimler”, “Türkçe Matematik”. Although grading system in most of the high schools are over 5, some high schools’ may be different, for example 10, so we needed to this field to specify this information. We think that the other fields’ meaning can be inferred from their names, and the previous tables’ explanations so we will not explain them.

HighSchool Info
• HSId *   • SchoolName *   • SchoolType *   • StartDate *   • GradDate   • GradingType *   • CGPA   • Branch *

Figure 2.27 – Structure of the "High school information" objects

Complete ER Diagram of the Job Seeking Web Sites – Job Seeker Part

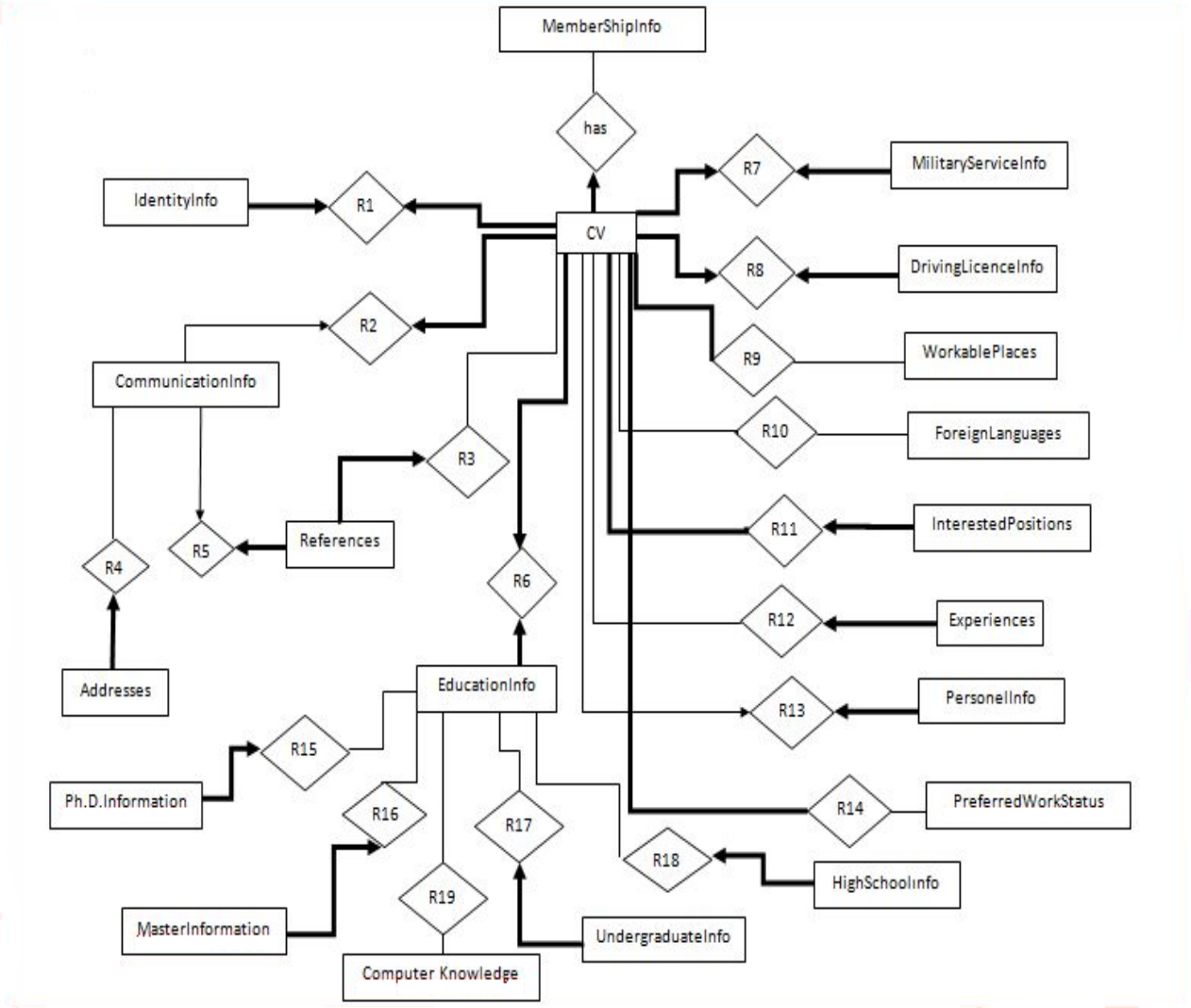


Figure 2.28 – Complete ER diagram of the JSWSs – Job seeker part

2.2. Data Objects of Job Seeking Web Sites – Employee Seeker Part

Employee Seeker Membership Information: This table stores all necessary information about member companies as our portal database do.

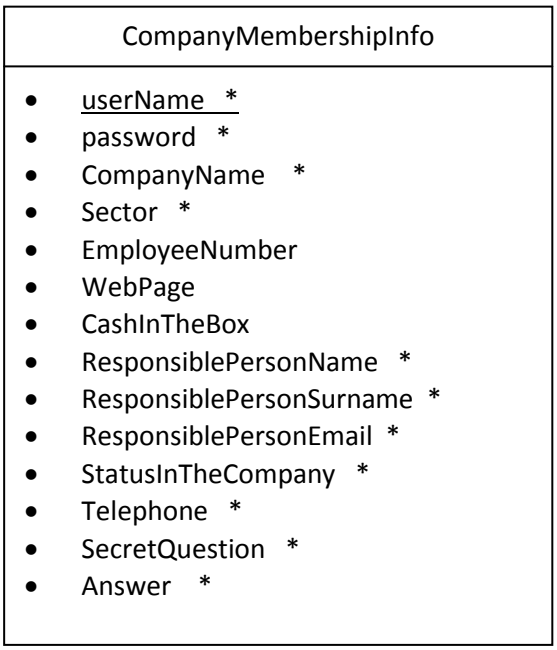


Figure 2.29 – Structure of the "Company Membership information" objects

Company Address Table: This table is not different from the "Company Address Table" of our portal's database. When a company wants to publish a job announcement, the form page includes all addresses of the companies as a list with radio buttons, and the responsible person will select the working place of the candidate employee from that list.

Figure 2.30 – Structure of the "Company Address" objects

Job Announcements: This entity stores all necessary information about a job announcement. When a company wants to publish a job announcement, the open position in the company must be described properly by filling the appropriate fields of the form in the "job announcement creation" page. On the other hand, the field which is named as "AdditionalJobInformationAsText" is for the company to describe the job specifications freely, and optional to be filled. The other fields includes little, specific information, and we will use those information in our web services to compare CV's and job announcements' similarities when a job seeking web site can not a find a suitable CV to a job announcement for a long time and wants help from our portal.

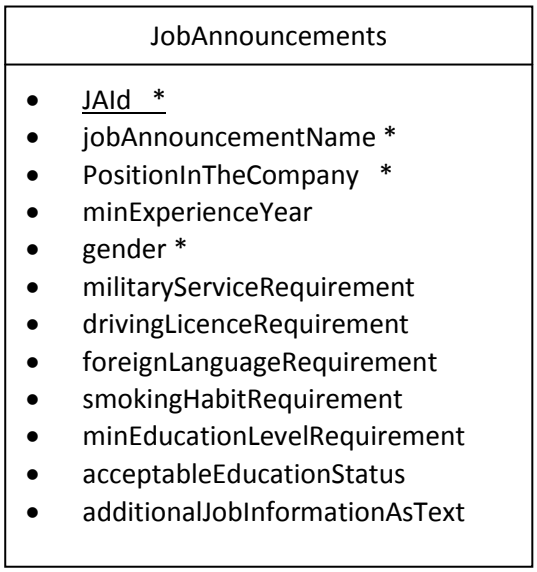


Figure 2.31 – Structure of the "Job announcements" objects

Job-CV Match Archive: After a company creates a job announcement and publish it in a job seeking web site, then candidate employees can leave their CVs to that job announcement. Company can look those CVs, and eliminate some of them, or all of them, or may select one for approval. When a company approves a CV then that job announcement will be dropped from job seeking web site's database and this CV-Job Announcement matching information will be stored to the table below. The information in this table can be drawn by "Match Report Web service" when statistical information is requested from our portal.

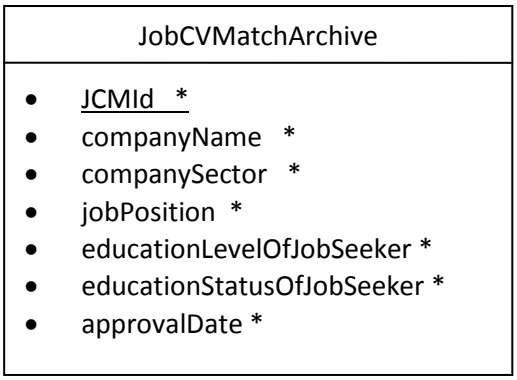


Figure 2.32 – Structure of the "Job CV match" objects

ER Diagram of the Job Seeking Web Sites – Employee Seeker part

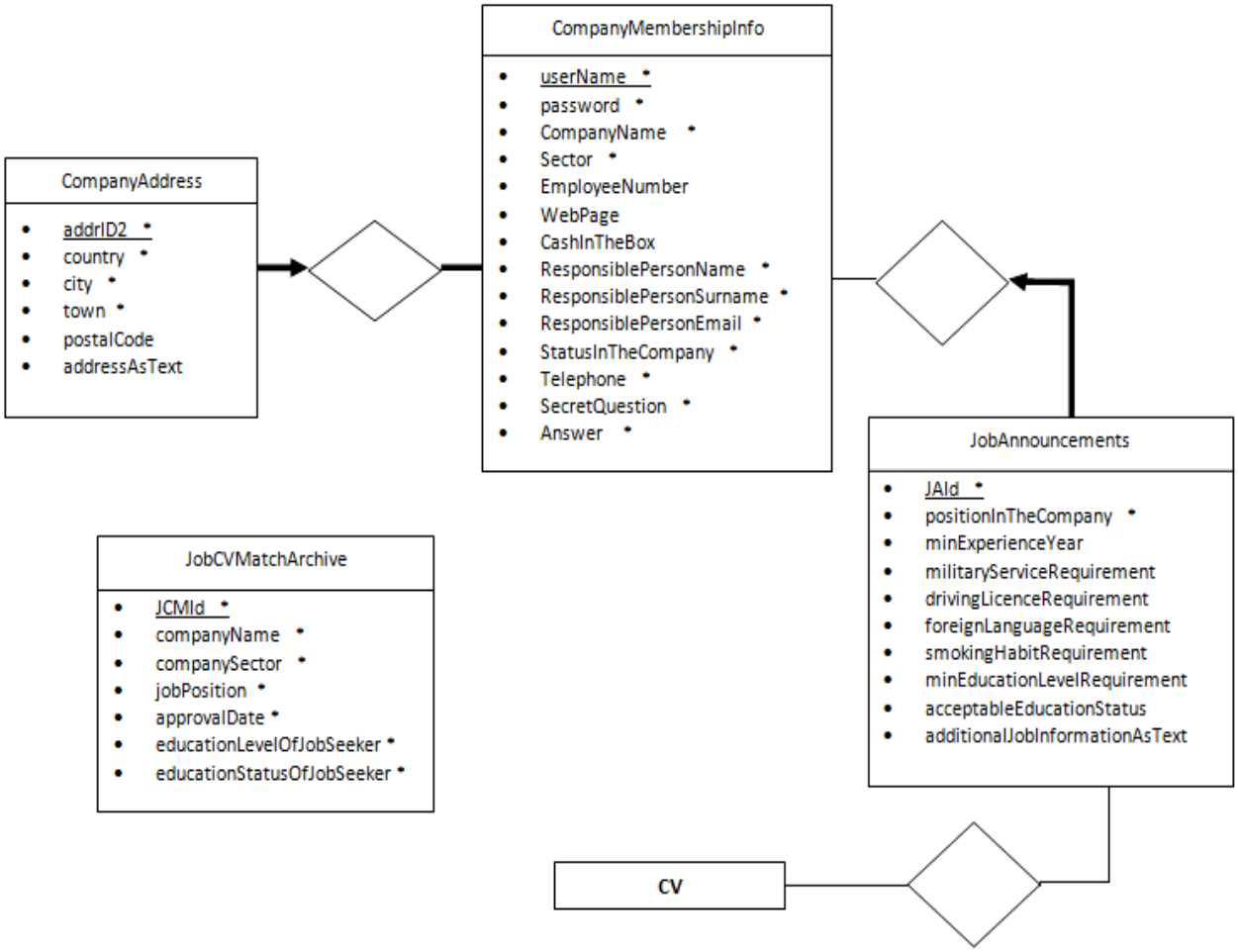


Figure 2.33 – Complete ER diagram of the JSWSs – Employe Seeker part

2.3. Data Objects of Our Portal

For our portal, there are two main objects “job seekers” and “employers” and their properties. Our website will be able to provide job seeking as a service open to everyone without requiring any registration steps. Anyone will be able enter our site and search for available jobs in categories. However, for those who are not registered to us, we will not provide any other service.

There will be two kinds of registration process: one for the ones who are seeking for a job, and one for the ones who are seeking for employees. Therefore, our two main data objects will be “Job Seekers” and “Companies”.

Job Seekers: We will only need some information of job seekers’ for their registration to our website. “Username” property for us to identify members, “password” for users to log in to the system, “secret question” and “answer” as usual for users to receive email when they forgot their password and request to learn what it was, one email address for us to send their passwords when they requests or send information about the jobs matching to them.

We are not interested in the job seekers educational or other personal information other than those we mentioned above. Such information is for job seeking websites to hold. The reason for why we will not hold such information in our database is we will not send CV and other information of the user every time that he wants to leave CV to any job application.

To clear this issue, let’s give an example. A user who has a member of our portal searched for job applications and suppose found an appropriate job for him and wants to leave a CV for that application. As mentioned in the system definition, he will not be able to apply that application from our site since the result-returning web site doesn’t let us to do so. We will only provide him a link to the related site. Therefore, we do not need to hold any information other than the attributes given below for job seeker users.

JobSeekerMembershipInfo
• Username *   • Password *   • Secret Question *   • Answer *   • Email *

Figure 2.34 – Structure of the "Job seeker membership information" objects

This will be our set of job seekers in our database. Notice that there are stars next to some attributes indicating those fields cannot be leaved as empty in the registration step.

Companies: Apart from the attributes in job seekers table, we need some additional information for companies membership. First of all, there must be a person in each member company who is responsible for employment issues of the company. That person should give a name, surname, e-mail address, and status of the himself/herself in the company, and a telephone number of the company for the person of our website, who is responsible for registration of the companies, to validate the membership request of the company. In other words, "ResponsiblePersonName", "ResponsiblePersonSurname", "ResponsiblePersonEmail", "StatusInTheCompany", which is also for the responsible person, and "Telephone" fields should be filled for registration to be completed.

Some other information about the company should also required like the name and the sector of the company.

Figure 2.35 – Screen shot taken from our Portal's user interface

Companies will be able to leave a job application from our web site as they can do from the other job-seeking web sites. Why we added such feature? The answer is actually simple, “more work in less time”. Suppose that an employer wants to leave an announcement that he needs an employee, however he is a member of more than one job seeking web site and he wants to leave the application to all of them. Then he has to log in to all those web site and fill the related fields again and again. However, with this feature of our portal, employers will fill the necessary fields, and will give the specifications they want only for once. Then by pressing “Gönder” button, the job application will be sent to all the chosen job seeking web sites.

If companies choose to send job applications to job seeking web sites by using our portal, the companies will not deal with depositing money to related job seeking web sites. We have a “CashInTheBox” field for each company. They will only deposit money to us and we will deal with payments to the other sites. They do not have to send money for membership, however, we require it before sending job applications to job seeking web sites.

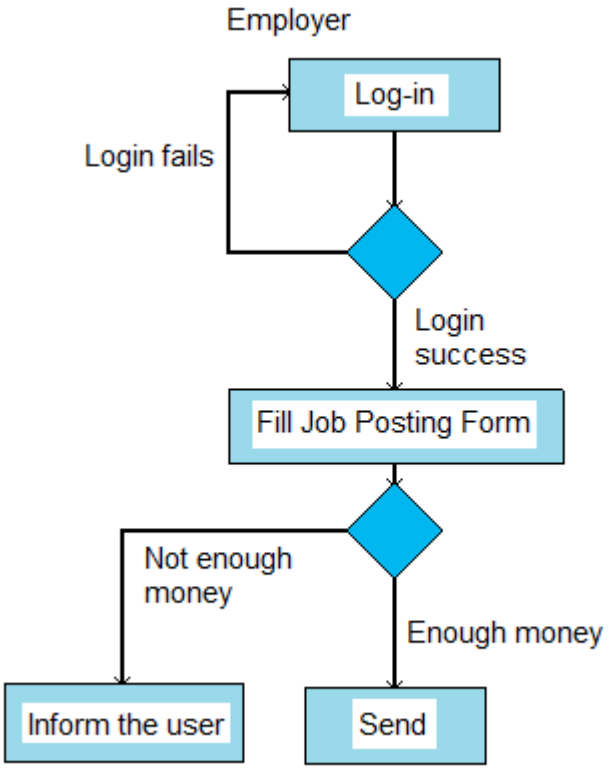


Figure 2.36 – Flow chart of the “Job announcement posting”

When an employer wants to send a job application to job seeking web sites, we first will calculate how much money needed to send the application form to all the selected web sites and check the status of the “CashInTheBox” of the company. If there is no problem about the money we will send the job application information and transfer the required amount of money to the selected sites and reduce the amount of the cost from the cashbox of the company. Otherwise we will inform the user about status of their cash box.

The attributes of the table holding information of the companies is given below. Remember that the stars next to the fields are indicating that those fields cannot be leaved as empty.

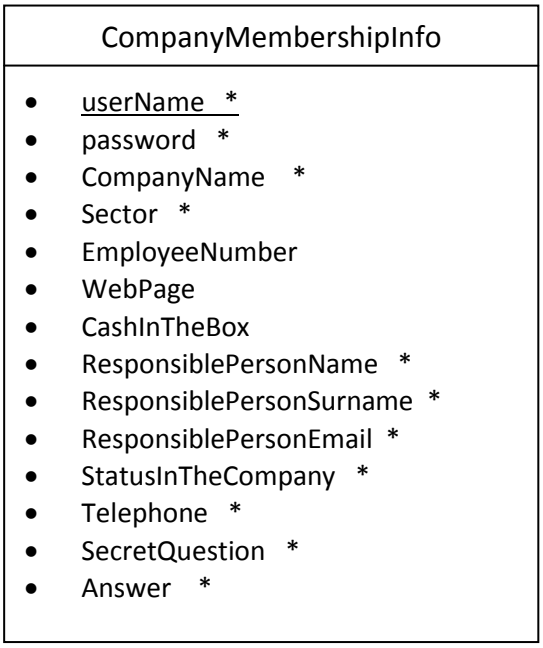


Figure 2.37 – Structure of the “Company membership information” objects

We decided to hold a separate address table instead of embedding an address field to Company membership information table. The reason for this is, a company may have more than one office in different locations.

Complete ER Diagram of our Portal

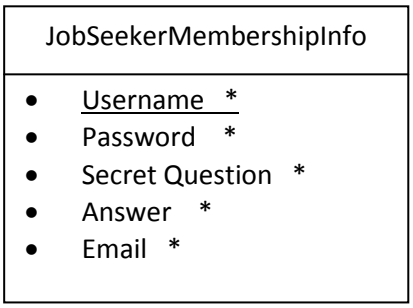


Figure 2.38 - a

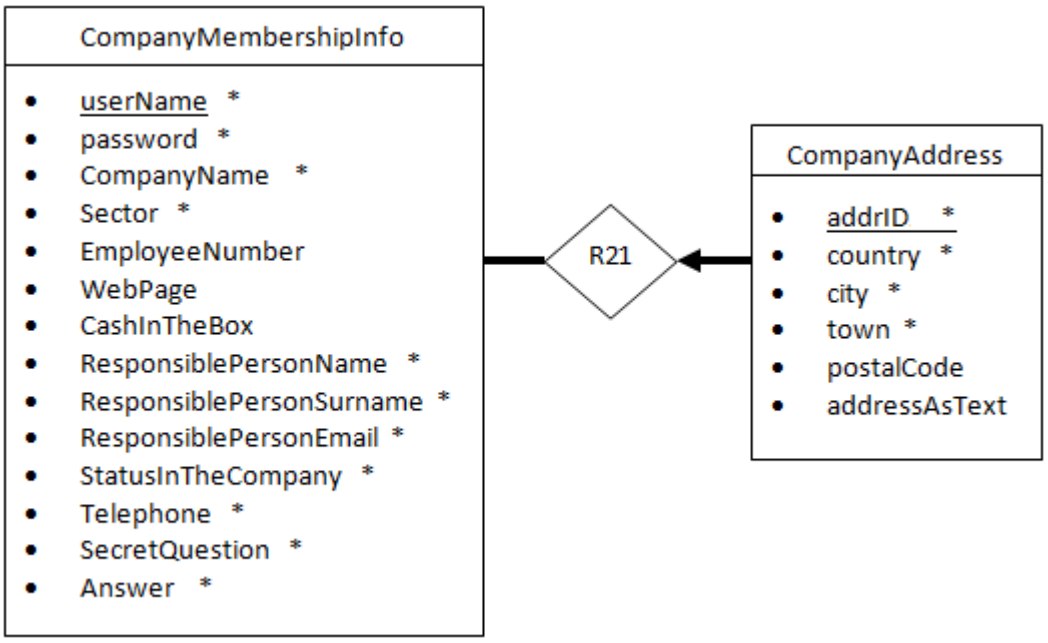


Figure 2.38 – b

PART III – ARCHITECTURAL DESIGN

3.1. DESIGN STYLE

We have chosen "**Layered Architecture**" style as we have three layers of our design.

- The outer layer is the "**presentation (user interface) layer**" running on the web browsers of the users of our portal.

- The middle layer is the "**Application layer**" running on the server of our portal. There will be two different class of application servers (servlets) :

    - The web services of our portal
    - The web services of Job seeking web sites

The first class of web services will run on the server of our portal. And the second class of web services will run on the servers of the job seeking web sites. Those web services will communicate via their "**client"** web services running on the opposite side, meaning that the client web services of the ones of our portal will run on the servers of the job seeking web sites and vice versa.

- The inner layer is the "**Data Storage layer**" running on the corresponding servers of the server applications. Data intensive web applications will insert, delete or fetch data objects from the databases of the systems that they belong to.

3.2. AN ABSTRACT LOOK TO THE SYSTEM MODULES

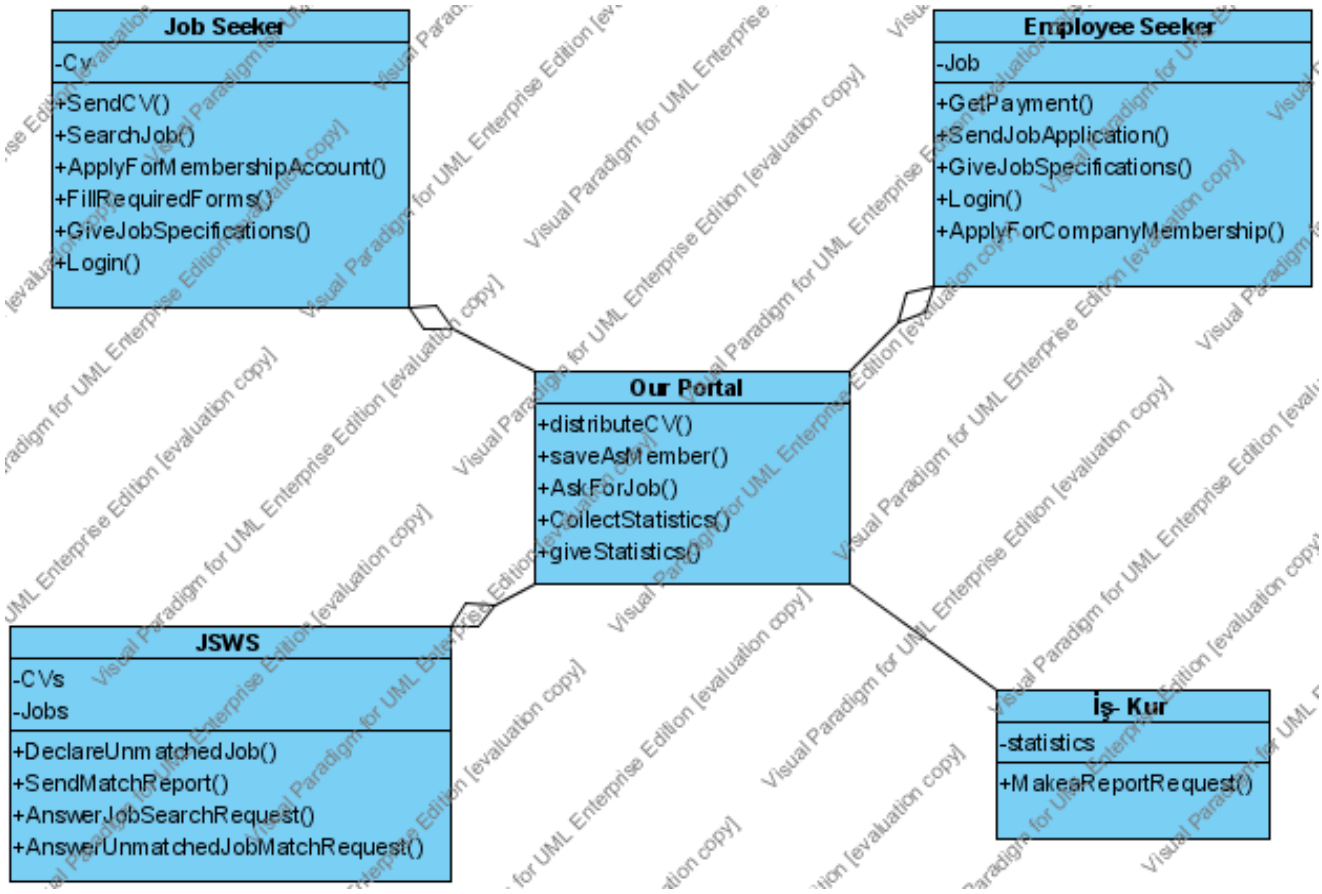


Figure 3.1 – An abstract look to the system

If we explain our class diagram, in the diagram there are five classes because there are 5 specific types.

All are connected with our portal because they can communicate through our portal not between each other. And it is shown clearly one to many or one to one relationships between them by using class diagram notations.

3.3. FLOW OF DATA BETWEEN THE SYSTEM MODULES

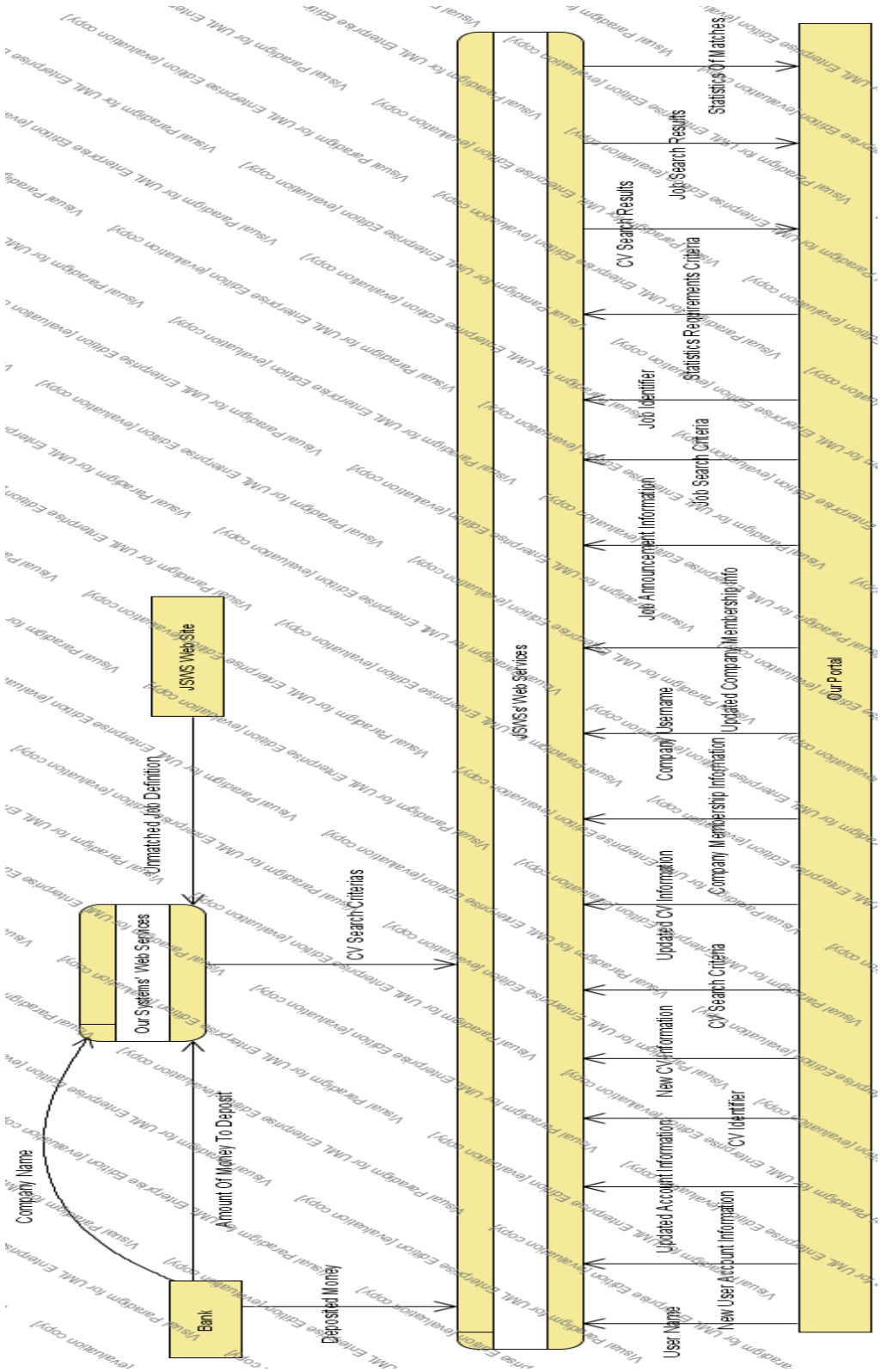


Figure 3.1 – Level - 0 Data Flow Diagram of the design

We have drawn our Data Flow Diagrams considering the web services we will use. This gave us the chance to represent the whole data flow all over the system. We have drawn two levels of data flow diagram, level-0 and level-1. The next level would be useless detail and would be difficult to represent without the level represented above. So we thought two levels of data flow diagram would be necessary. We defined the inputs and outputs of the system by looking at the web service definitions we made. While writing the inputs and outputs, sometimes we grouped similar data under a name. This is because that, writing each input and output name would make our data flow diagram a mess. Also on level-1, for readability reasons, we divided the data flow diagram into two pages. One shows the diagram about the web services on JSWS side and the other page shows the ones about our portal. The notations of the dataflow diagram object may be a bit different from the most commonly used way to show dataflow diagrams. This is because that our diagram drawing tool, Visual Paradigm, has a notation as this. In these diagrams, the processes are represented by a partitioned rectangle. The rectangles without any partitions show the external entities of the system. The rectangle with a 'D' inside shows the data store. We include these in order to avoid any confusion when compared to the traditional way of drawing Data Flow Diagrams.

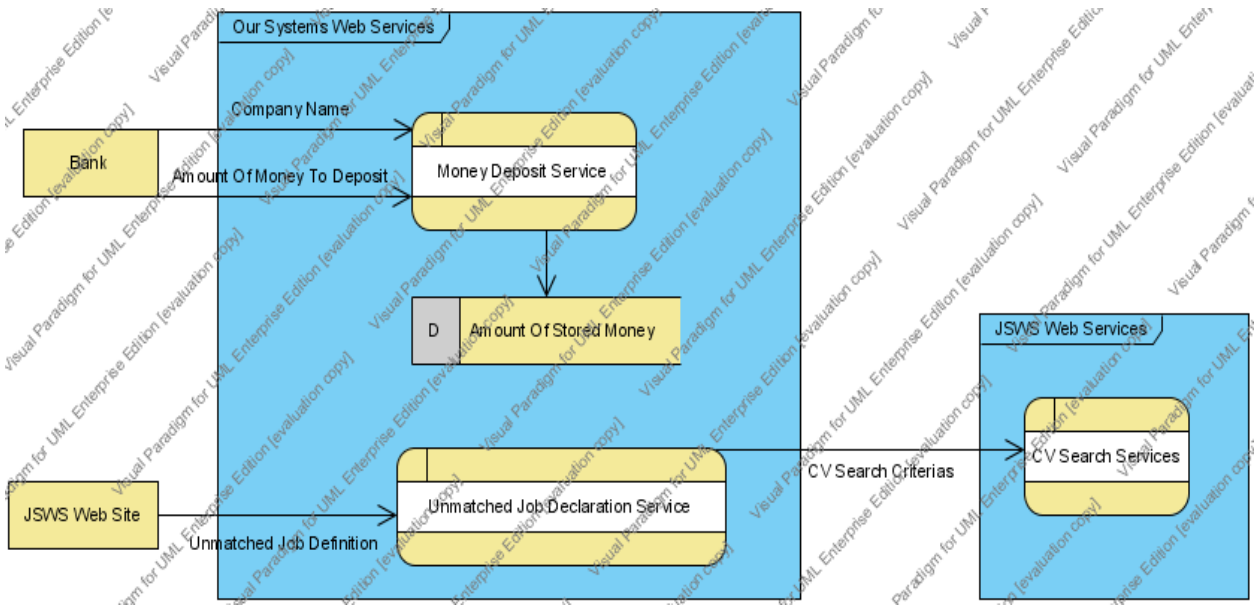


Figure 3.2 - Level – 1 Data Flow Diagram within our Portal

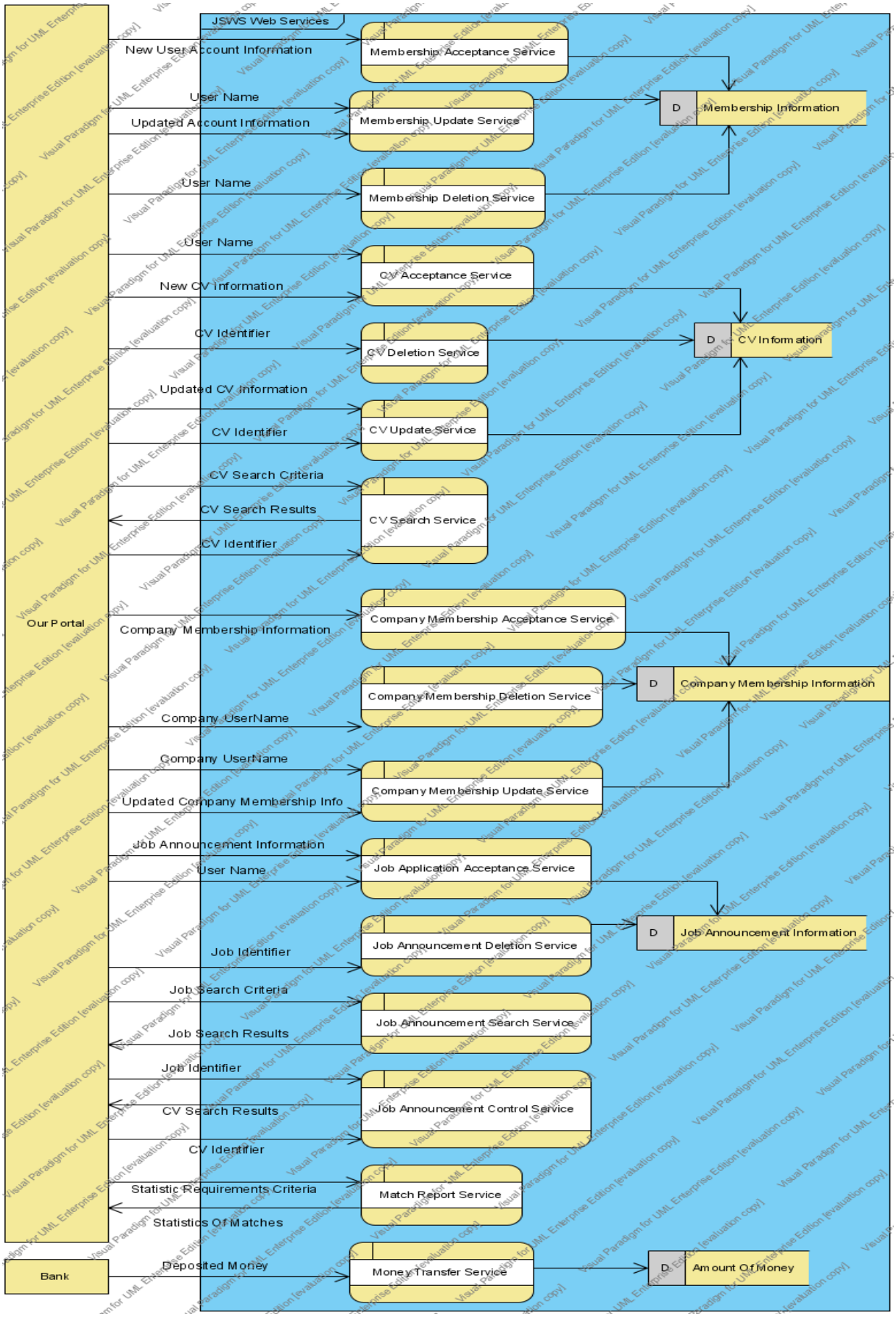


Figure 3.3 - Level – 1 Data Flow Diagram within JSWSs

3.4. SCENARIO BASED MODELING

In our initial design report, we tried not to change our use case diagram since. But we needed two minor changes in the use case diagram.

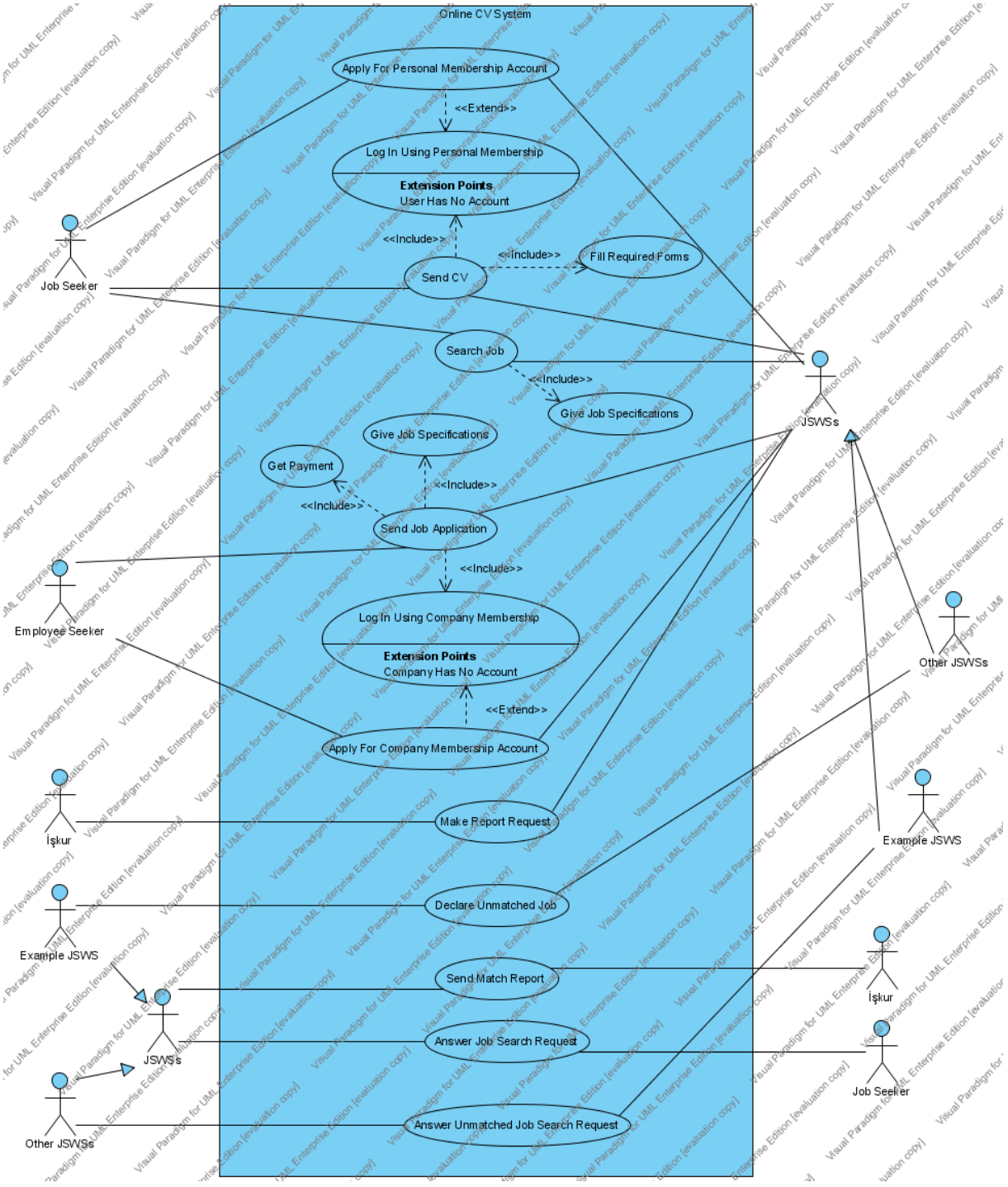


Figure 3.4 - Use case diagram

After the discussions we have made, we found out that there was no need to put an unmatched CV declaration service. This is because that the problem is todays world is not lack of CVs but lack of jobs. There we found out that this kind of service would never be used. This kind of service would also mislead the company about the CV and they may not want to give a job to such a CV. So we removed that service from our use case diagram.

The second minor change is that the name “Clients” for job seeking web sites became out of date after the development we had. So we will use the name “JSWS” instead of client actors afterwards. These names are updated from our use case diagram.

3.5. SEQUENTIAL BEHAVIOUR OF THE SYSTEM

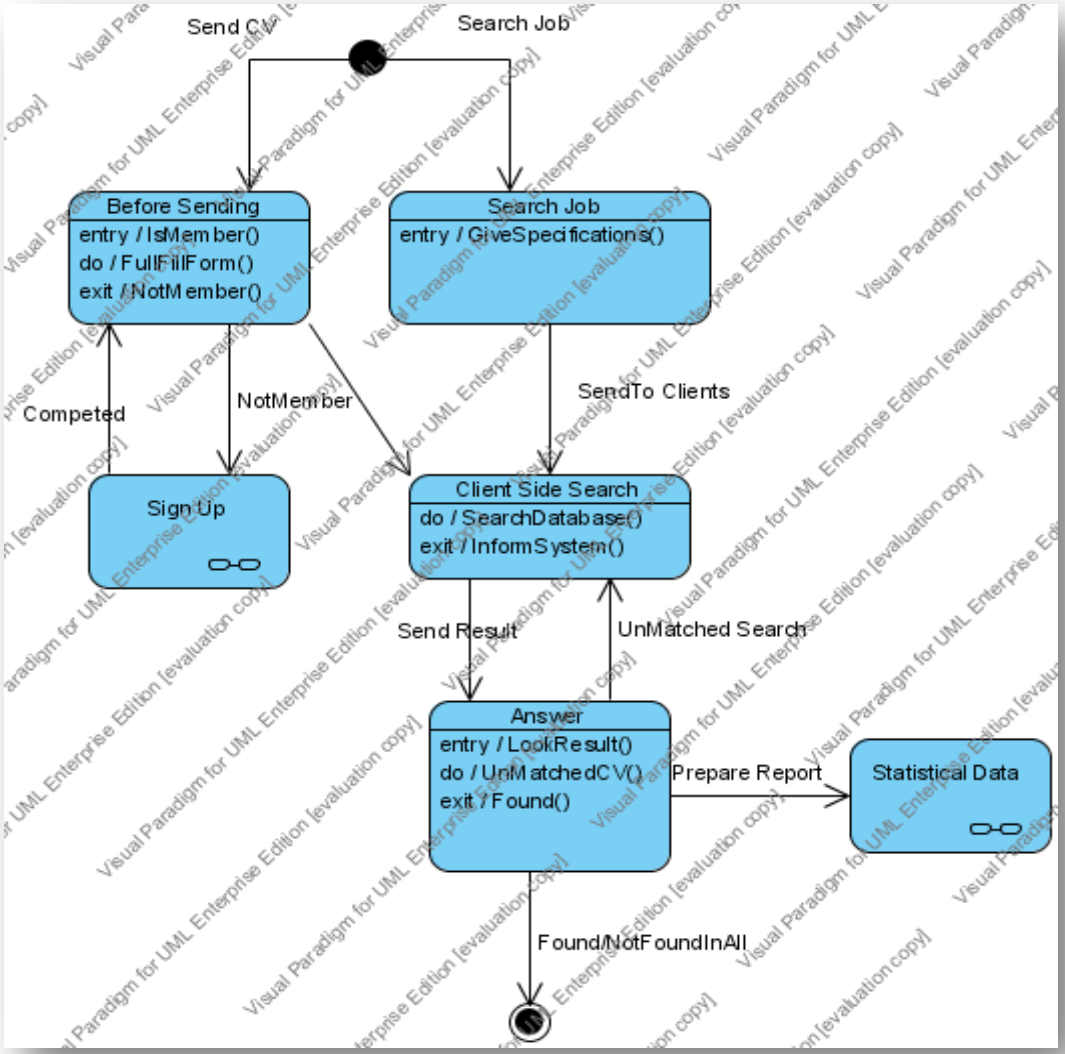


Figure 3.5 – Sequential behaviour of our system – part 1

First state diagram is based on the services “Send CV” and “Search Job”. These are the services are for job seekers. This state diagram shows that:

If a job seeker wants to send a CV to clients through our portal, first he has to be our member. Then he will full fill the form for CV and it will be distributed to the clients. On the other hand, to search a job, he has to give the specifications and then according to these specifications, it will be searched in the databases of the clients. Looking at job and CV matches, it can be prepared statistical report to İş-Kur which desires such report .If a CV is not matched in its’ own site then it will be in the status of an unmatched. Then it will be searched in the other sites. This system is also valid for the unmatched job.

Second state diagram is based on the service “Send Job Application” for the employee seeker. Employee seeker must be a member of the system and he has to give the payment before sending the job application. If he is not a member, he has to be sign up the system. After completion of the sign up operation, he can return to the sending job application. During the state “before sending”, he has to full fill the job application form. This form will be sent to all the clients. At the state “Declaration in Clients”, it will be put all the job applications to all of the clients .After this operation completed it will be finish.

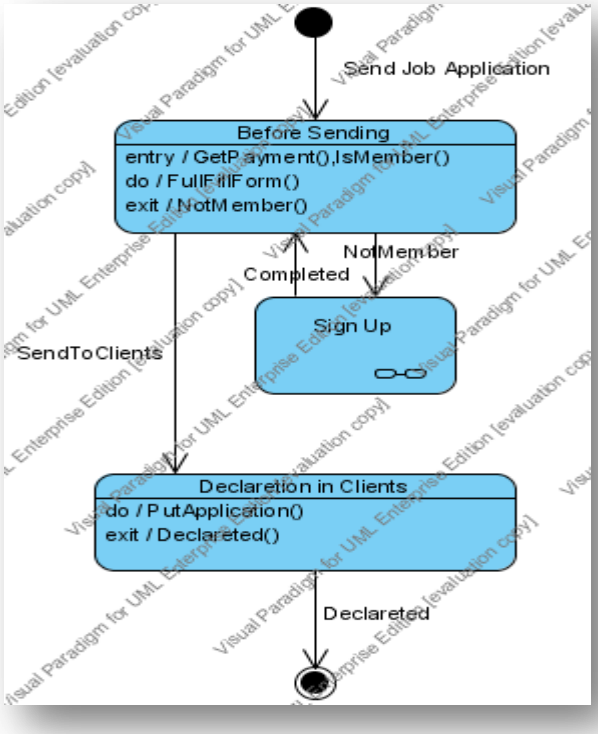


Figure 3.6 – Sequential behaviour of our system – part 2

PART IV – USER INTERFACE DESIGN

The most important point in user interface design is the ease of use of the system. The user should easily understand how the system works by just looking at the user interface. This easiness shoul d result with the easiness of usage. While using this system, user shouldn't memorize what he did before. The steps should be leading. Another requirement for user interface design is that all the properties of the system should be reached with out trying to find how to reach a service of the system. Finally, users want beatiful looking web sites consisting of attractive colors and and tools.

These are all known issues of designing user interfaces. Considering these, we have developed a prototype of our web portals web site which allows users to use our system more easily. We have used Dreamweaver in these designs since we had problems with application designer and couldn't use it. We have used javascript in order to add dynamic content to our web page. By using javascript, we have made the most important part of our web page, the tabs. Usage of tabs provide the issues mentioned above and makes the web page easier to use. All parts of the system can be easily found by the usage of these tabs. They also help us to create more delightful web sites.

Since we are designing a web site for job seekers and employee seekers, we felt that the colors should create a web site which is elegant. The word elegant created the color grey in our mind. So we have made our color decisions on different levels of grey. To help user find something easier, we tried to keep our pages as small as possible. We think that this design decision will make our web site interesting and easy to use by the users who use our portal.

During the design of our user interface, we learned many things about javascript and other tools for designing web. This prototype is really basics of the web page at the moment. We will certainly develop these pages more and we ensure you that it will be a lot better in the next steps of our project. We prepared this prototype just to show you the main design we made in our mind. So there may be some changes in our design after a while. But for the moment this is the main layout of our web page.

Figure 4.1 - Homepage of OnlineCV

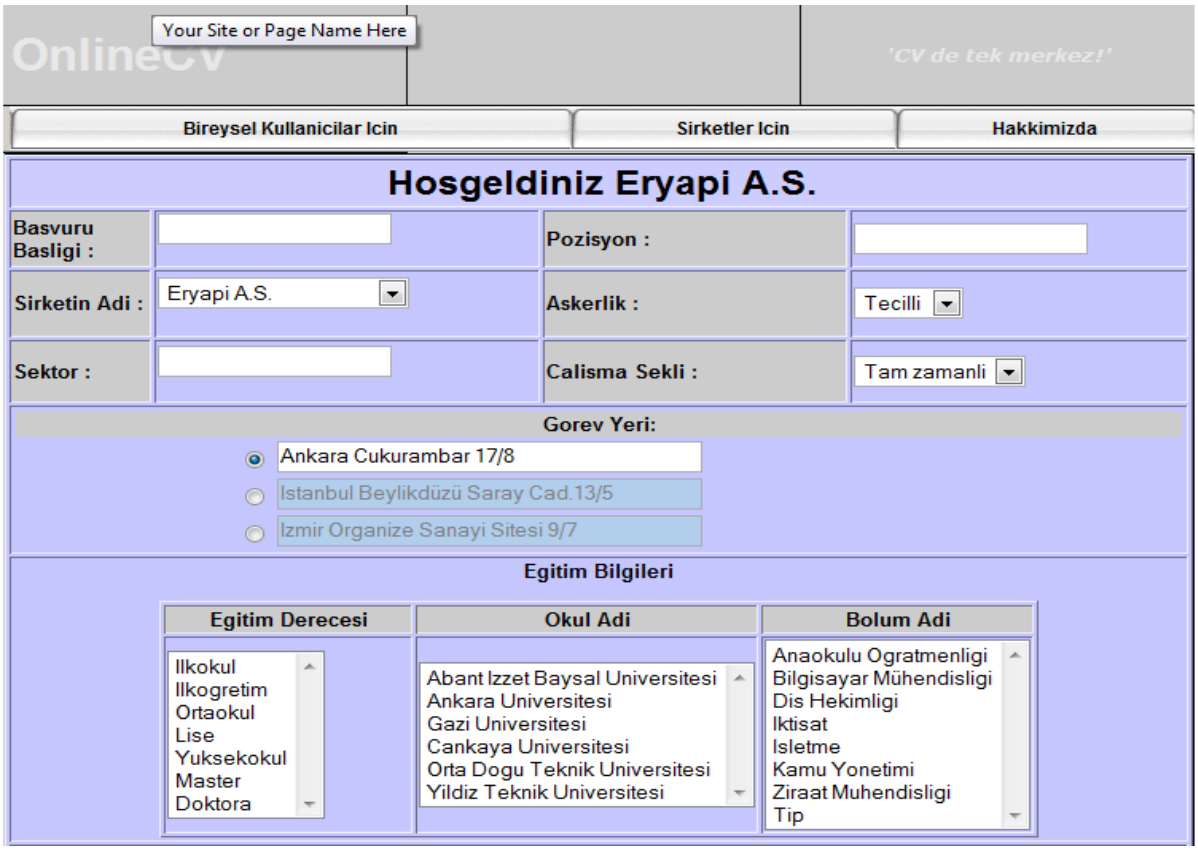


Figure 4.2 - Fill the "Job Announcement form" – part 1

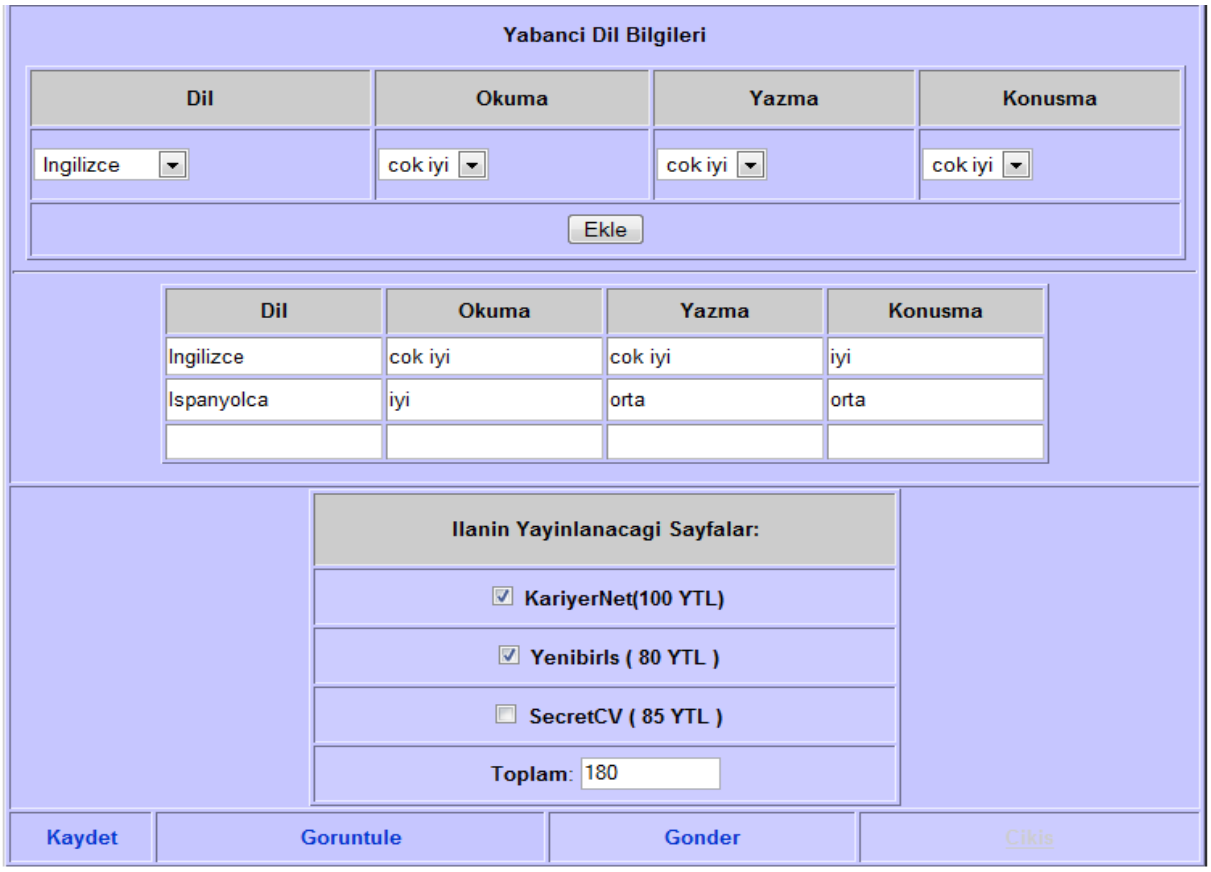


Figure 4.3 - Fill the "Job Announcement form" – part 2

PART V - PROCESS SPECIFICATIONS & WEB SERVICES DESIGN

In this part of the report, we will define and describe the web services we require from the "Job Seeking Web Sites" (in short, we will use the notation JSWS for these web sites) and the web services we provide to those web sites and the bank web site. We first want to mention about the web services that those web sites should provide us. Then we will describe the web services provided by our portal.

5.1. Design issues of the sub-modules (WEB SERVICES)

5.1.1. Web Services of the Job Seeking Web Sites

Membership Acceptance Service: This service will give us the chance to create a new membership on the JSWS for the person who uses our portal. When someone creates a membership from our portal, we will use this service of JSWS to create the same membership in them. This service consists of two methods.

bool usernameControl(**string** name)

We are going to use this method to check whether that the given username exists or not. As a result of this check, the method will return us a boolean value. Depending on this return value, we will inform the user about the availability of the username he/she specified.

bool insertUserInfo(**string** [] informations)

After all the fields are filled by the user in the registration step and "Gönder" button is pressed, our portal will send the information filled by the user to the JSWSs. This method will get the message sent from our portal and will insert the user to its database. If somehow an error occurs while the insertion step, this method will return us a fault message and we will inform the user about the failure. Otherwise, if no problem occurs we will insert the user to our portal and inform the user about the success.

Membership Acceptance Service
- **bool** usernameControl(**string** name)
- **bool** insertUserInfo(**string** [] informations)

Membership Information Update Service: This service will be used for User Account Information updates in JSWSs. When a user updates his/her account from our web site, we

have to send these updates to JSWSs in order to avoid confusion between our portal and them. This update process will be handled by this service. This service contains three methods: one for changing the password, one for changing the secret question and answer, and one for changing the email. The username won't be able to be changed after the creation of the account. Because, this field uniquely identifies the users.

bool changePassword(**string** username, **string** newPasswd)

When a user changes password in our portal, we will invoke this method and send the new password to the JSWSs as a string. This method will receive the message from our portal and try to change the password of the username and return successfulness of the process.

bool changeSecretQuestionAnswer(**string** username, **string** newQuestion,
string answer)

Similar to changePassword method, when a user changes the secret question and the answer to this question, we will invoke this method. But this time we have two string parameters to send. One is the new secret question and the other is the answer to this question. Depending on the values of these variables, the old question and the answer is replaced with these new values. Finally, it will return us whether or not the operation succeeded.

bool changeEmail(**string** username, **string** newEmail)

The old email address of the user is overwritten in all the JSWSs by the use of this method as someone changes his/her email address from our portal, and this method will return us a message about the accomplishment of the task as the other methods do.

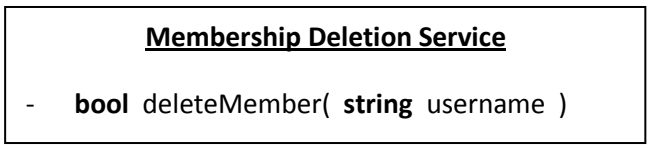
Membership Information Update Service

- **bool** changePassword(**string** username, **string** newPasswd)
- **bool** changeSecretQuestionAnswer(**string** username, **string** newQuestion , **string** answer)
- **bool** changeEmail(**string** username, **string** newEmail)

Membership Deletion Service: This method is used by our portal for deleting a member from JSWSs. When a user wants to delete his membership from our portal, we invoke this web service in order to accomplish the deletion task. This web service needs only one method:

bool deleteMember(**string** username)

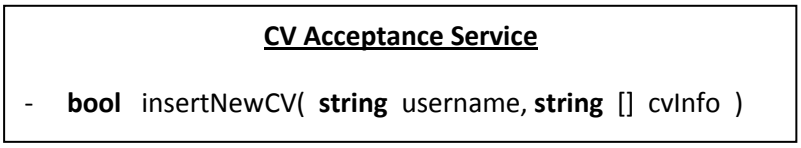
This method is called with the parameter "username" of a member. When this method is called for one of the JSWS, the JSWS should delete the membership without any requirement. However, if somehow any error occurs, it will return us the failure.



CV Acceptance Service: A member can create a new CV through our portal at any time he/she wants. By the usage of the username and additional information that user inserts by filling a form, we send the resulting CV to JSWSs and the "insertNewCV" service method of the JSWSs will insert the CV of the member to the related database.

bool insertNewCV(**string** username, **string** [] cvInfo)

This method is called with the two parameters. The username and all the CV information as a string array is send to JSWSs, and "insertNewCV" method will insert the newly created CV and returns a boolean value to our portal indication the successfulness of the insertion operation.



CV Deletion Service: A user may want to drop his/her CV whenever he/she wants. This reverse process can be done by CV deletion service of the JSWSs. We require two different methods for this aim.

bool deleteCV1(**string** username, **string** cvName) or

bool deleteCV2(**int** cvID)

Either cvID or username and cvName can be used for uniquely specifying a CV. Therefore, we can use one of the methods above in the implementation phase, for deletion of the chosen CV. The JSWS should send us feedback about the successfulness of the process.

CV Deletion Service
- **bool** deleteCV1(**string** username, **string** cvName)
- **bool** deleteCV2(**int** cvID)

CV Search Service: If a job announcement is not applied any job seeker for a specific time period, then the JSWS, in which that job announcement is published, may use "Unmatched job declaration service" of our portal. In this case, we will need to make a search among all the JSWSs' CV data objects. In order to accomplish this task we require JSWSs to provide us a CV Search Web Service. This service consists of 4 methods.

int [] returnCVsWithTheseCriteria(**string** [] criteria)

We will call this function with a string array consisting of criteria of the corresponding job. After calling this, the JSWS will make a search from its own database and return the ID's of the matched CV's (if any).

int returnCVId(**string** username , **string** cvName)

This function will be used to get the CVId by giving the owner name of the CV and the title of the CV.

string [] returnCVOwnerAndTitle(**int** cvId)

This is the inverse of "returnCVId". When our portal needs a to get the username of the CV owner and the CV title, this method will be invoked.

string [] returnDetailedCVInfo(**int** cvId)

This method will be called when the details of a CV is required. By giving the CV Id, we can request the JSWS to return the details of the CV related with the Id given as parameter.

CV Search Service

- **int** [] returnCVsWithTheseCriteria(**string** [] criteria)
- **int** returnCVId(**string** username , **string** cvName)
- **string**[] returnCVOwnerAndTitle (**int** cvId)
- **string** [] returnDetailedCVInfo(**int** cvId)

CV Update Service: This service will be used when a user want to change a part or all of his/her CV. There are seven methods for updating a CV. For the attributes belonging to the CV itself, we have direct changing methods. If there are other entities connected to the CV, we decided to use the CV deletion and insertion services sequentially. So we have a function for

replacing the old information with the new one. Let us begin with the methods changing a specific attributes of CVs.

bool changeCurrentWorkStatus(**int** CVid , **string** newStatus)

If the user's working status changes after the creation of the CV, the user will be able to change the work status part. For example, when someone leaves his job, he will change the working status part of the CV with the "update CV" option in the homepage. When a member updates only this part of his/her CV this method will be called by our portal. We will send the new working status as a string and will wait for the respond. Finally, we will inform the user about the result.

bool changeJobExperience(**int** CVid , **int** experienceYear)

If the user's work experience in years change in a way, this change can be done on the CV entity by the usage of this method. The new experience year is sent as a parameter to the method and the change is accomplished on the CV defined by CVid.

bool changeActiveStatusOfCV(**int** CVid , **bool** isActive)

The CVs can be in active mode or passive mode. Companies can only see the active ones. This situation can be changed depending upon the user's choice. A user can change the active status of his/her CV through our portal so our portal requires this method from JSWSs. We change the status of the attribute by sending a boolean and the CV Id to JSWS.

bool changeCVLanguage(**int** CVid , **string** newLang)

The language that the CV is presented is defined as a CV attribute. For example, if the language is set to English, the job experience area is named as "Work Experience" but if it is Turkish it is named as "İş Deneyimi". To change language attribute, we invoke this method. The language the user wants to change is sent with the "newLang" parameter as a string. We again wait for the respond of the method.

bool changeCVName(**int** CVid , **string** newCVName)

The user may want to change the name of his/her CV. For this aim “changeCVName” is called just like other previous methods and the attribute holding the CV name is changed.

bool changeAdditionalCVInfoAsText(**int** CVid, **string** infoAsText)

As the other parts of the CV, when the user wants to change this part, this method will be called by our portal.

bool replaceCVWithNewCV(**int** oldCVId , **string** newCVInfo)

For all other kind of changes, an entity connected with CV entity should be changed. Therefore, the change is done as a whole using this method. New CV information is send using this method and the entity related to that information is changed. And this method will invoke CV deletion service and CV insertion service sequentially.

CV Update Service

- **bool** changeCurrentWorkStatus(**int** CVId , **string** newStatus)
- **bool** changeJobExperience(**int** CVId , **int** experienceYear)
- **bool** changeActiveStatusOfCV(**int** CVId , **bool** isActive)
- **bool** changeCVLanguage(**int** CVId , **string** newLang)
- **bool** changeCVName(**int** CVId , **string** newCVName)
- **bool** changeAdditionalCVInfoAsText(**int** CVid, **string** infoAsText)
- **bool** replaceCVWithNewCV(**int** oldCVId , **string** newCVInfo)

Company Membership Acceptance Service: Similar to job seeker membership acceptance service, this service provides methods to add a company user to the JSWSs. There are two methods for this aim.

bool usernameControl(**string** name)

This method checks whether the specified user name for the company membership exists or not. Depending on the value returned, a decision is made about inserting the membership or not.

bool insertMembershipInfo(**string** [] informations)

If the usernameControl returns true, this means that the username is available and the company membership may be created. The required informations are sent as a string array with this method and the company account is created. If the account cannot be created, false is returned, else the return value of the method is true.

Company Membership Deletion Service: The company may be closed or may not want to work with some of the JSWS anymore. At this situation, the membership of the company should be deleted. We do this by calling the following method.

bool deleteMemberCompany(**string** username)

Given the user name of the company, this method deletes the company membership from the JSWSs.

Company Membership Deletion Service
- **bool** deleteMemberCompany(**string** username)

Company Membership Information Update Service: There are many things that can change in a company. These changes should cause our system to be out of date about the information

of the company. So we require a company membership update service from JSWSs. Many of the details of a company can be changed using these 11 methods.

bool changePassword(**string** companyUserName, **string** newPasswd)

Because of some security reasons, companies may want to change their password. This can be accomplished by our portal with the "changePassword" method call. Given the company username and the new Password this method saves the new password for the company as the password of the company.

bool changeSecretQuestion_Answer(**string** companyUserName,
string newQuestion , **string** answer)

Again for some security reasons, the company may want to change the security question and answer stored for them. Similar to changing password, they can change this information stored in the JSWSs by using our portal. Our portal does this update by the use of "changeSecretQuestion_Answer".

bool changeResponsiblePersonInfo(**string** companyUserName, **string** name ,
string surname , **string** email, **string** status , **string** telephone)

The person who is responsible from human resources may be changed as a result of change in company positions. This will be corrected using "changeResponsiblePersonInfo" method by our portal. The information details about the new responsible person are sent with string parameters of the method and the required update is done on the company defined by the "companyUserName".

bool changeCompanyName (**string** companyUserName, **string** companyName)

Company may change only its name although its place, sector and other information remain the same. Updating company name is done from our portal with "changeCompanyName" method call.

bool changeCompanySector(**string** companyUserName, **string** sector)

The company may change its sector and start working on another sector. This change is reflected to the sector attribute of the company with this method. The new sector is sent as a string to the JSWS. The change is done on the company defined by the "companyUserName".

bool changeEmployeeNumber(**string** companyUserName, **int** number)

The company may fire some people or take new people as employees. This affects the information on the employee number attribute. To update this attribute we require this method from the JSWS web site. Similar to methods defined previously the update is done by giving the number of employees to the method. The update is done on the company account defined by the company user name.

bool changeWebpageURL(**string** companyUserName, **string** newUrl)

When a company changes the URL of their web site and inform us about this, we invoke this method to update the webpage URL information from JSWSs. The new URL is sent with the companyUserName and the required update is done in JSWSs.

bool addNewAddress(**string** username, **string** [] addressInformation)

The company may open a new branch or a new office in a different address. This new address information can be added by the usage of this method.

int [] getAddressIdsOfTheCompany(**string** username)

When we need the addresses of a company, we invoke this method giving the company user name as the username. This method returns us an integer array of id numbers

of addresses of the company. By using these ids, we can get the detailed address information by the next method.

string [] getAddress(**int** addressId)

Given an address id, this method returns the detailed information about the address as a string array. This array consists of the attributes of address entity.

bool deleteAddress(**int** addressId)

Given an address id, this method deletes that address from company addresses entity. This may be called as a result of a closed branch/office of a company or closing of a company itself.

Company Membership Information Update Service

- **bool** changePassword(**string** companyUserName, **string** newPasswd)
- **bool** changeSecretQuestion_Answer(**string** companyUserName, **string** newQuestion , **string** answer)
- **bool** changeResponsiblePersonInfo(**string** companyUserName, **string** name , **string** surname , **string** email, **string** status , **string** telephone)
- **bool** changeCompanyName (**string** companyUserName, **string** companyName)
- **bool** changeCompanySector(**string** companyUserName, **string** sector)
- **bool** changeEmployeeNumber(**string** companyUserName, **int** number)
- **bool** changeWebpageURL(**string** companyUserName **, string** newUrl)
- **bool** addNewAddress(**string** username, **string** [] addressInformation)
- **int []** getAddressIdsOfTheCompany(**string** username)
- **string []** getAddress(**int** addressId)
- **bool** deleteAddress(**int** addressId)

Job Announcement Acceptance Service: In order to insert job application to JSWSs, we use this web service. When a company selects some of the JSWSs, this service is invoked and the job announcement is sent to be published in those JSWSs.

double getAmountOfMoney(**string** username)

This method is used to determine how much money that the company deposited to chosen JSWS. The user may send money to the JSWSs through banks without any relation with our portal; the JSWS will keep the amount of money deposited in its database. This method will let us see that money. The amount of money deposited apart from us is returned as a double value. By looking at this value if the amount of money is enough to leave a job application, we will call the following method directly. If not, we will look at the amount of money that the company has in our portal. If this amount completes the required money to post a job application we will send the missing amount to the JSWS and then call the method below. If the sum is not enough, we will warn the company to deposit some money.

bool insertNewAnnouncement(**string** username, **string** [] jobAnnInfo)

If the conditions mentioned in the method above are satisfied, the job application details are send as a string array. The job application is then posted in the JSWS web site.

Job Announcement Acceptance Service

- **double** getAmountOfMoney(**string** username)
- **bool** insertNewAnnouncement(**string** username, **string** [] jobAnnInfo)

Note that after a job announcement is sent it will not be updateable. Before sending the announcement we will present the user how the announcement will look like and sending the announcement after seeing the appereance means accepting the appeareance of the job announcement with its contents. Therefore, after sending the announcement, we will not

provide the employee seeker companies to change what they have sent. Because we cannot control whether the user has made so many changes that the announcement he/she sent has evolved to a new job annoncment, or not. Users should not be able to send new announcements by updating the old ones they have already paid for.

Job Announcement Deletion Service: The company may find an employee for the job application published before and may want to drop it. This is accomplished by two methods in different ways.

bool deleteJobAnn1(**string** username, **string** jobAnnName)

By this method, our portal will delete the job announcement using the username of the responsible person and the job announcement name. Because these two field uniquely identifies the job announcements. By logging in to our portal and giving the application name, the application will be deleted from JSWS invoking this method on them.

bool deleteJobAnn2(**int** jobAnnId)

Another way that our portal can use for deletion of the job announcements is identifying them by their Ids. Our portal will invoke this method to delete the given application from JSWS.

Job Announcement Deletion Service
- **bool** deleteJobAnn1(**string** username, **string** jobAnnName)
- **bool** deleteJobAnn2(**int** jobAnnId)

Job Announcement Search Service: People may search jobs from our portal and see the results without any registration step. From different JSWSs we will gather the appropriate job

announcements according to the given search specifications. This is accomplished by the use of Job Announcement Search Service. There are 4 methods to do this search from our portal.

int [] returnJobAnnsWithTheseCriteria(**string** [] criteria)

The user may select some criteria from our web site and click "Search" button. This button will invoke a method in our portal which calls the method above of the "Job Announcement Search Web Service" and send the criteria's that the user selected as a string list to JSWSs. The results are returned giving the job announcements' Ids as an integer array.

int returnJobAnnId(**string** username , **string** jobAnnName)

To search for a job with a given name and company, our portal will invoke this method on the JSWSs' side and get the job application id as a return value.

string [] returnJobAnnInfo(**int** jobAnnId)

This method is the reverse method of the "return Job announcement Id". The owner's username of the job announcement and the title of the job announcement will be returned.

string [] returnDetailedJobAnnInfo(**int** jobAnnId)

This method will be used when the user who makes the job search, requires detailed information about one of the results. The method is invoked with the job announcement id and the result is returned as a string array consisting of the details of the given job announcement.

Job Announcement Search Service

- **int []** returnJobAnnsWithTheseCriteria(**string []** criteria)
- **int** returnJobAnnId(**string** username , **string** jobAnnName)
- **string []** returnJobAnnName(**int** jobAnnId)
- **string []** returnDetailedJobAnnInfo(**int** jobAnnId)

Job Announcement Control Service: This service is about the relationship between CV and Job announcements. There are four methods to manipulate these relations in JSWS side. These are as follows:

int [] returnCVIdsOfJobAnn1(string username, string jobAnnName)

When the user wants to see the CVs that applied for a specific application, we use this method and the method below to find who applied for that job. This method will return an array of CV Ids.

int [] returnCVIdsOfJobAnn2(int jobAnnId)

As mentioned above, this method does the same thing with the method above but this time the search is done by announcement id.

bool deleteCVfromJobAnnList(int jobAnnId , int cvId)

This method is called by our portal when a CV is eliminated by a company from its job announcement list.

bool approveCVInJobAnnList(int jobAnnId , int cvId)

If a CV is accepted by the company, which gives the job announcement, all other CV's that applied to it will be dropped when our portal calls this method.

Job Announcement Control Service

- **int** [] returnCVIdsOfJobAnn1(**string** username, **string** jobAnnName)
- **int** [] returnCVIdsOfJobAnn2(**int** jobAnnId)
- **bool** deleteCVfromJobAnnList(**int** jobAnnId , **int** cvId)
- **bool** approveCVInJobAnnList(**int** jobAnnId , **int** cvId)

Match Report Service: Since we will provide some statistics to some foundations about job – CV matching status with some criteria, we will require the JSWSs to provide us a service namely "Match Reporting Service" that sends statistical data to our portal. This service has two methods.

int returnMatchesWithTheseCriteria1(**string** [] criteria, **Date** dateAfter)

With this method, we can require the matched job announcements and CVs for a given criteria after the date given as parameter. For example, we can require the matches done in last 5 days on health sector. This method returns us the number of matches.

int returnMatchesWithTheseCriteria2(**string** [] criteria, **Date** dateBefore)

With this method, we can require the matched job announcements and CVs for a given criteria before the date given as parameter

Match Report Service
- **int** returnMatchesWithTheseCriteria1(**string** [] criteria, **Date** dateAfter)
- **int** returnMatchesWithTheseCriteria 2(**string** [] criteria, **Date** dateAfter)

Money Transfer Service: This service is used both by our portal and the bank to add money to the account of a company in JSWSs. After adding the required money to their cash box the company can leave their job announcement. The only method of money transfer service is given below.

bool addMoneyToCompany(**string** username , **double** amountOfMoney)

This method adds specified amount of money to the company account associated to the username.

Money Transfer Service

- **bool** addMoneyToCompany(**string** username)

5.1.2. Web Services of our Portal

Money Deposit Service: By the use of this service our portal will add money to the company account. The user of this service will be the bank. There is only one method to do this.

bool addMoneyToCompanyAccount(**string** username, **double** amountOfMoney)

This method deposits the amount of money defined by "amountOfMoney" to the company account defined by the "username".

Money Deposit Service

- **bool** addMoneyToCompanyAccount(**string** username , **double** mountOfMoney)

Figure 5.1 - A prototype user interface for money depositing from banks

Unmatched Job Declaration Service: This service will be used by the JSWSs. They will use this service when no CV is applied for job announcements for a specific time period. By this service we will search other JSWSs CVs database and we will try to find a match for the given job.

string [] findCVsforMe(**string** jswsName , **int** jobAnnId)

The job application that couldn't be matched is defined by the "jobAnnId". And the "jswsName" defines the JSWS that unmatched job announcement is held. So we make the search from other JSWSs and try to make a match.

Unmatched Job Declaration Service
- **string** [] findCVsforMe(**string** jswsName , **int** jobAnnId)

NOTICE! Employee seekers can not search CVs through our portal. Since such service would make us a rival to Job seeking web sites, we have not included it. Our aim is not to be a rival to other job seeking web sites but to help them

5.2 Control Specification

In general this diagram will show you an overall look to the portal we have designed. In our system, through our main web page, we can reach some sub pages or sub states. The state of the system is changed by the user interactions like clicking on buttons. When a button is clicked, a new page will be opened for example. After work in a state has been completed, the page will be closed and the page that has been traversed before will be opened or the main page will be loaded.

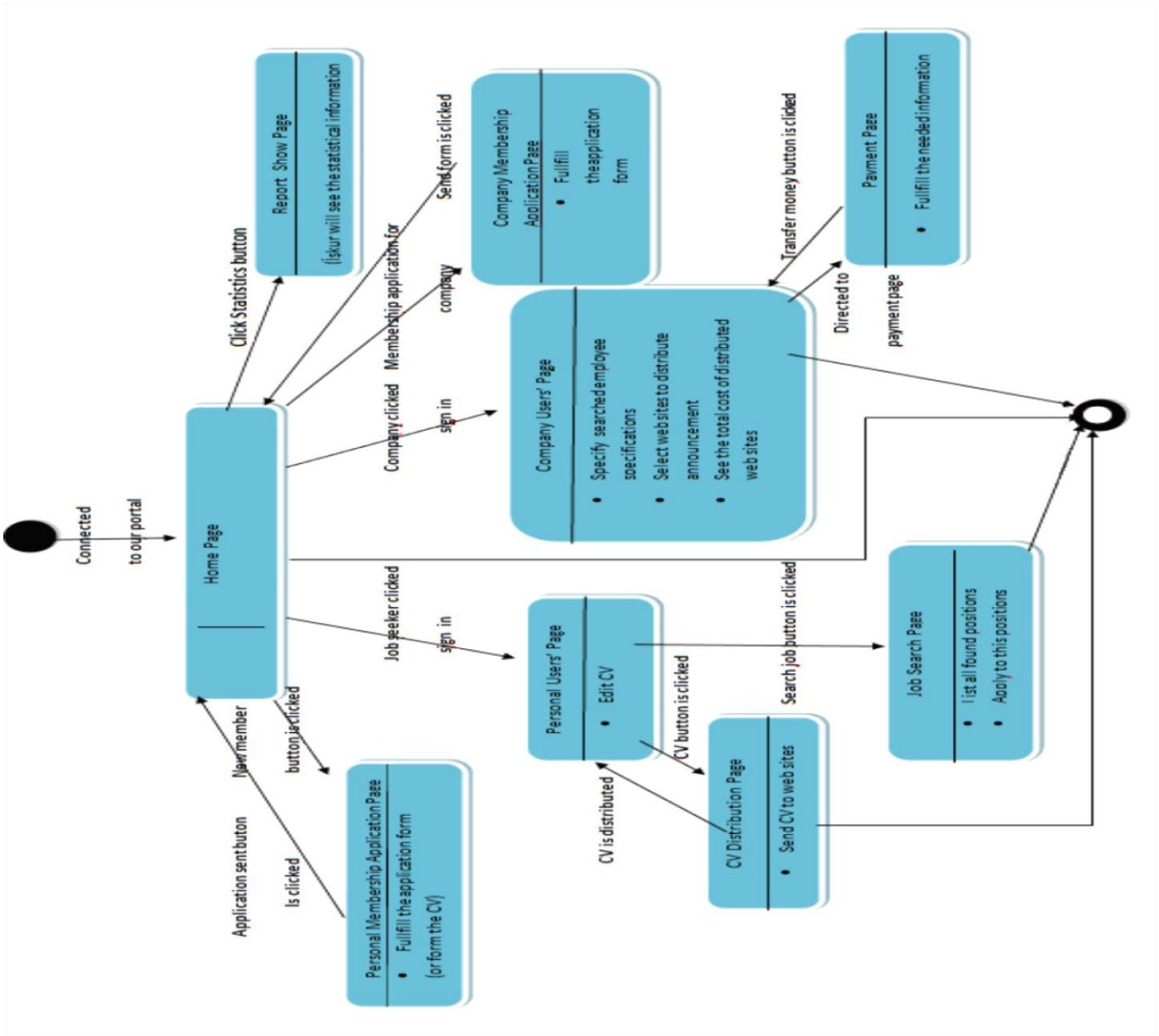
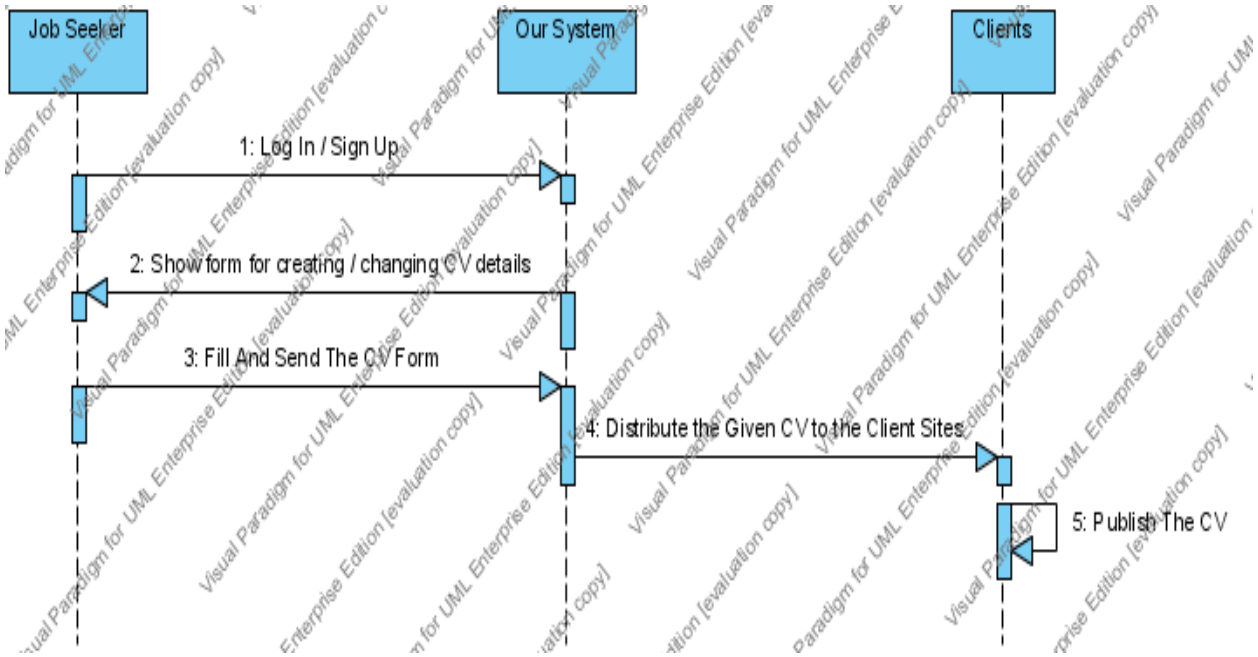


Figure 5.2 - Control Diagram of the Portal

5.3 DETAILED SEQUENTIAL LOOK TO THE SYSTEM in DIFFERENT SCENARIOS

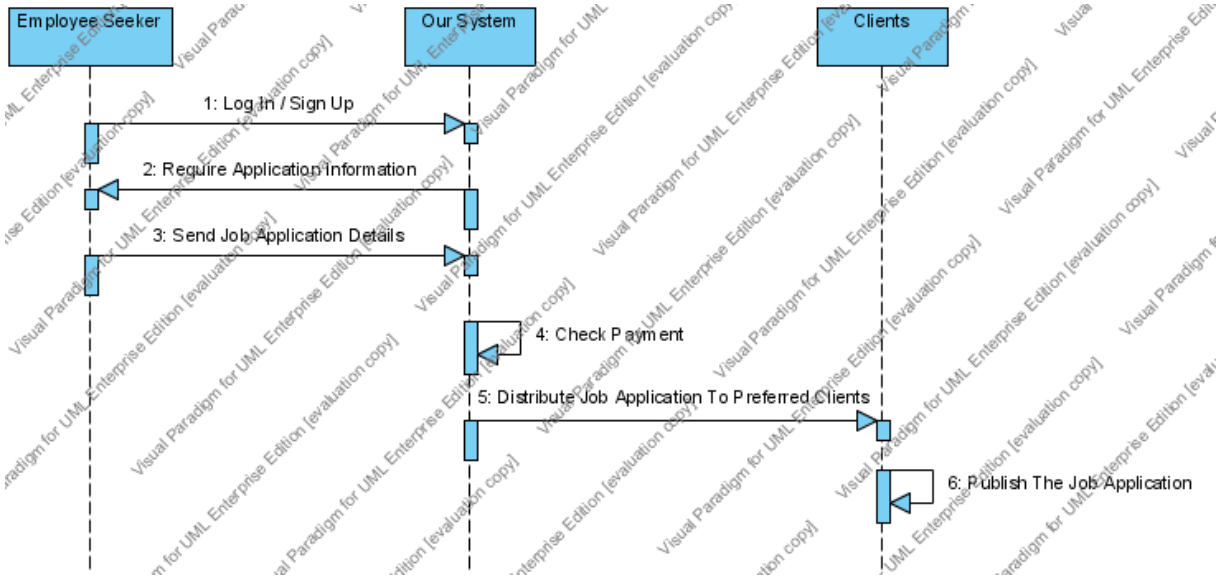
CV Distribution Diagram: A job seeker wants to leave his/her CV.



First our system has to identify the person who wants to leave CV. Therefore, a “Log In” process is required and if the person has not registered to our system yet, then he/she has to sign up. Then our site will show a menu from where people can create their CV or can make changes on their CV that they have created before.

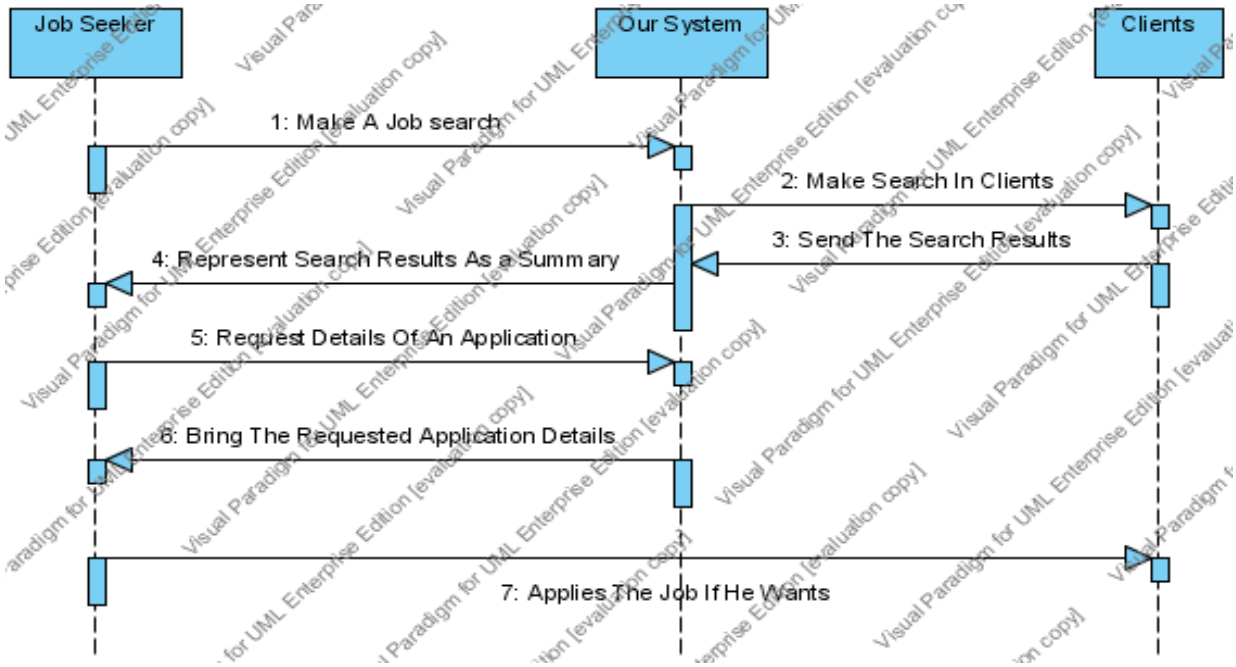
After the job seeker fills the form, our portal will distribute the newly created or updated CV to all the Clients.

Sending Job Announcements: A person, who needs employees, wants to make an announcement about a job.



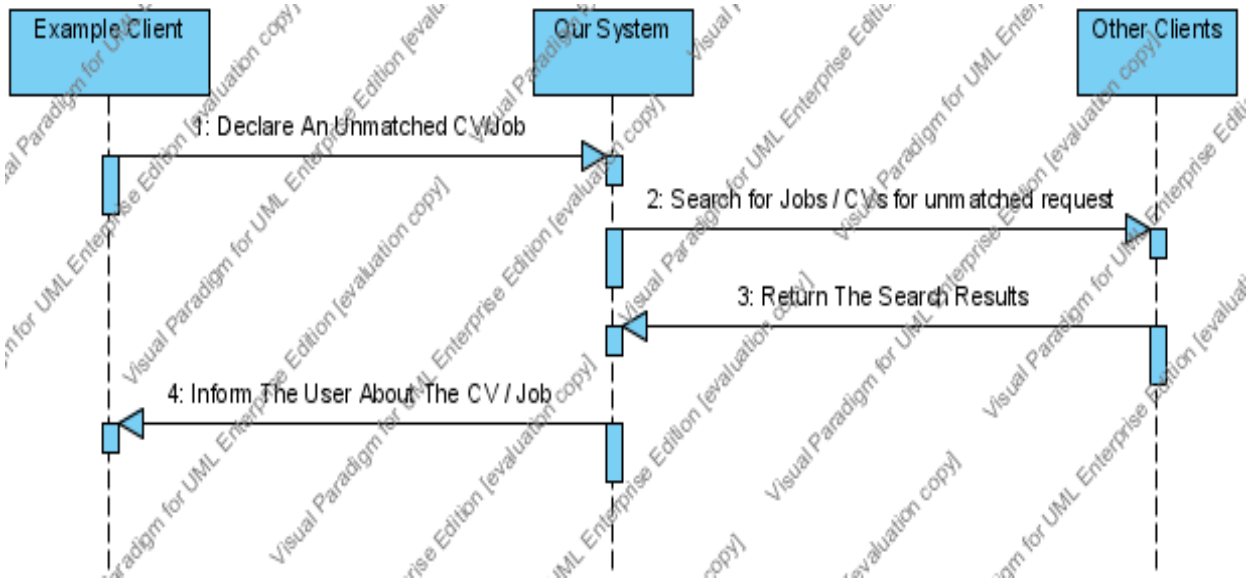
Employee Seeker has to log in to our system. And the ones who enters into our system for the first time, has to sign up. Our system will request detailed information about that job. After the employee seeker sends the details to us, our system will check the payment and if there is no problem about it, our portal will send the job application to all the clients that the employee seeker have chosen for publishing.

Searching For Jobs: A person looks for a job suitable for him/her.



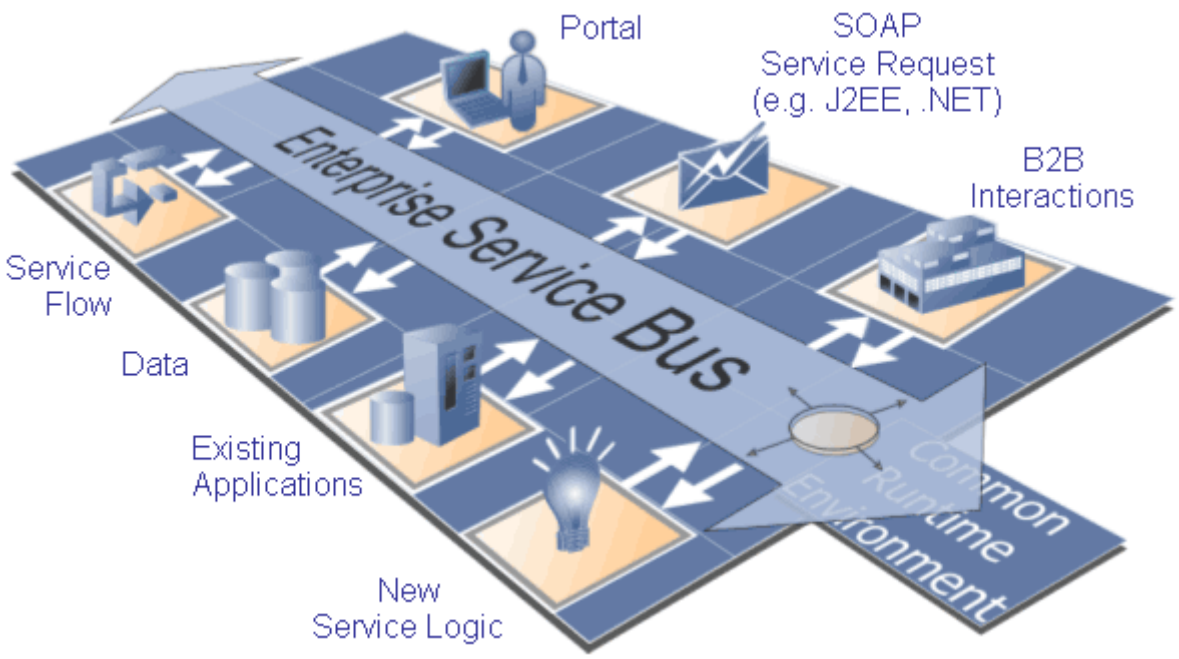
Job seeker can search for jobs without logging into our system. When a person requests for a job search, our portal will send a message to the clients including the specifications of the job to be searched. The web services of the clients will send us back their search results and we will combine them and present to the user. User will be able to see the detailed information about any job in the result table and will be able to apply if he/she wants.

Unmatched CV/JOB Declaration: A client wants to send us unmatched CVs or Jobs to us so we can search from other clients to match them.



Clients declare us an unmatched CV or Job. Then we send request to other clients so that they search their database and send us their result. Finally, we inform the client about result.

5.4. INTEGRATION ISSUES of the SERVICES & ENTERPRISE SERVICE BUS (ESB)



While the integration of applications is the biggest difficulty, by means of ESB, that integration can be done quickly and in a cost-effective way. With this technology, we can adopt a more modular development in our applications, and then we can integrate our distributed modules rapidly. This modular development approach allows us to create and use well-defined, reusable, and secure components, so that the risks in a development process can be minimized too. In addition, when our needs change as the time passes, ESB provides us the flexibility to adapt our systems to the new requirements easily.