PseudoSoft

CENG-491

DETAILED DESIGN REPORT

YENILINK PROJECT

Assistant: Ali Orkan Bayer

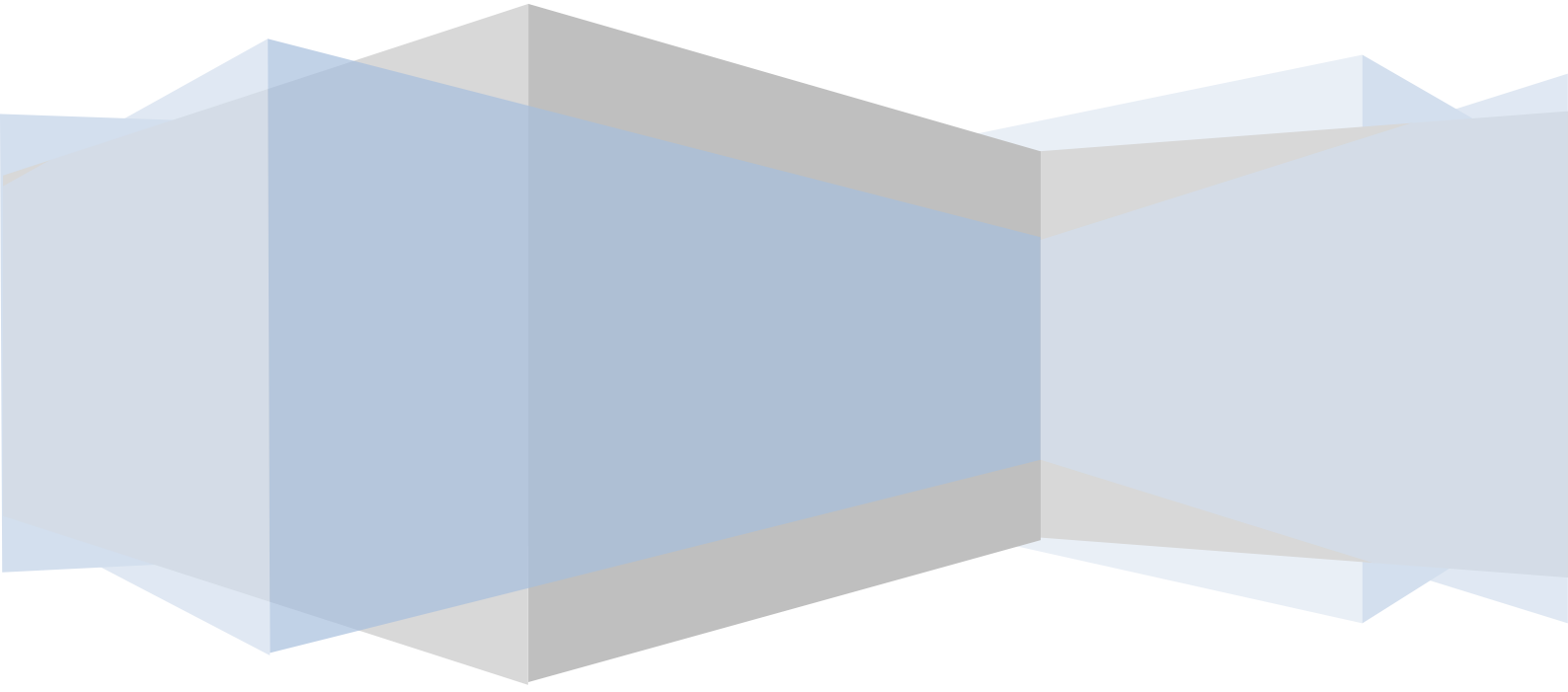


Table of Content

Continues ...

GROUP MEMBERS

Name	Number	E-mail
Furkan Kürşat Danışmaz	1394881	k.furkan@gmail.com
Ömer Nebil Yaveroğlu	1449248	omernebil@hotmail.com
Mehmet Bahattin Yaşar	1395664	e1395664@ceng.metu.edu.tr
Gülsüm Selcen Mülazımoğlu	1395276	selcen.mulazimoglu@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. "**Roles & Scenarios**" gives an overall look to the scenarios based on different roles. Finally in the introduction part, "**Tools we are using**" gives information about the tools we are using to implement the project, why we choose them and how we are going to use them.

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. We specified how we are going to solve the problem created by the different database structures of different Job Seeking Web sites.

PART III – ARCHITECTURAL DESIGN

Architectural design includes our **design sytle** and **an abstract view** of our **system modules** with the explanations of our decisions. After giving an abstract look, we gave a detailed look to each service of the system and showed the flow of data between services and their clients. "**Data exchange between system services (messaging)**" part explains the importance of reliable messaging and security issues. Finally in this part, "**Scenario based modeling**" explains who or what effect which modules in our system and what results occur.

PART IV – USER INTERFACE DESIGN

The user interface part will show you how our portal will look like.

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.

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, YENILINK. 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, we 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 a job seeker creates a CV in our portal, we distribute that CV to all other websites immediately if they are available. If they are not, timer will trigger a module in our system to try to send the unsent CVs again, until no unsent CV remains. Although we store all

the CVs and the job announcements in our portal's database, we also send those information to the other websites to store. 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 deposit webservice of our portal, they can send money to us. To be able to do this, we added 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.

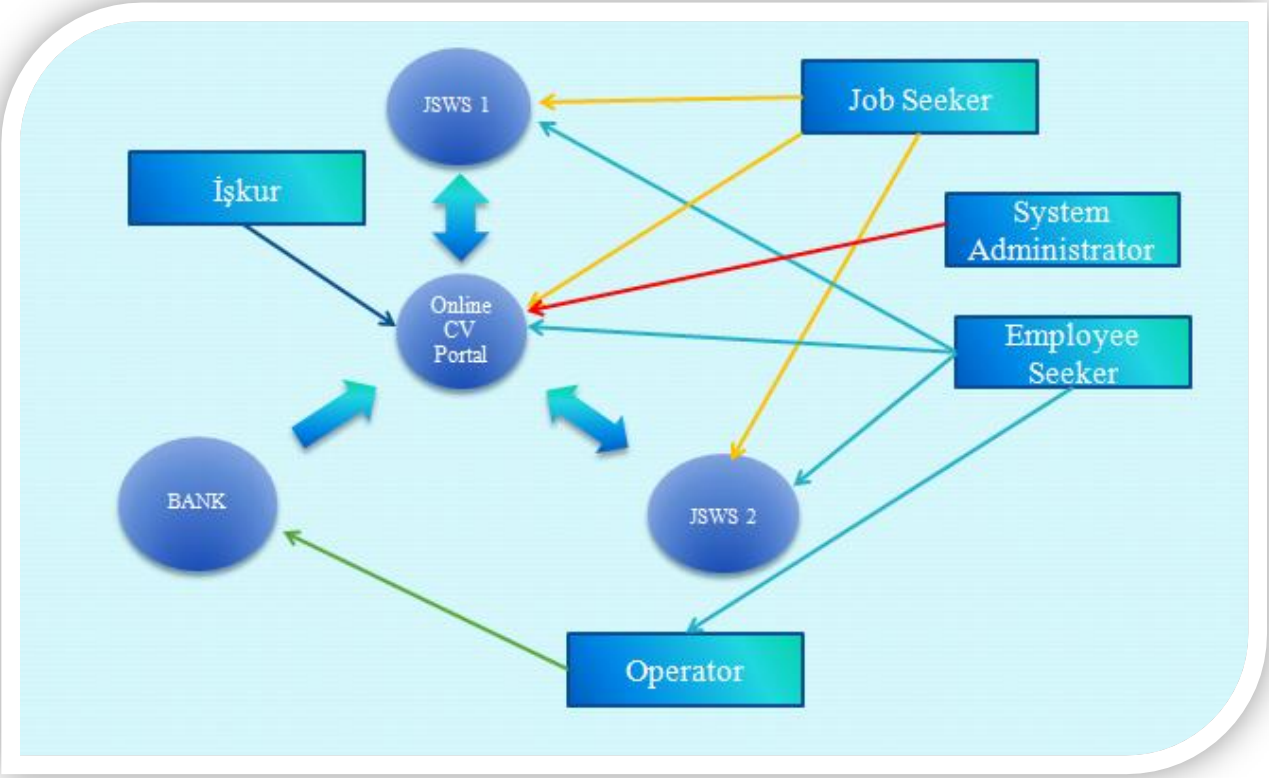
During those interactions with our portal's users, we will try to make things as simple as possible. In addition to the services which we provide to employees and companies, we will also provide a service which presents statistics about employee acceptance with graphics. To provide such a service, our portal will use the "Job-CV match information service" of all the 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. (By the way, from now on we will use JSWS abbreviation for Job seeking Web sites)

As you can predict from the explanation of our system, our portal requires to use all JSWSs' 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. PROPERTIES OF THE SYSTEM(ROLES & SCENARIOS)

To explain what the system provides for different roles, we created the graphic below which gives an overall look to all system.



Connections between System Roles & Modules

Role – 1, namely a "job seeker", can create a membership account in our portal and after that create CV for himself/herself. At any time the created CVs can be edited or deleted. As anyone can do, a job seeker can also search for job announcements by specifying some criteria.

Role - 2, namely employee seeker, can create a membership account for the entire company where he/she works for. At any time he/she can create a Job announcement for the company, can delete a previously created one or manage the CVs that applied to a job announcement belongs to that company.

Role - 3, namely İş-kur can query our system to get information about the employment of the sectors with any criteria they want.

Role - 4, namely our administrator of the portal, as a final role he/she can manage the database tables, can edit any of them.

There is also a role related to our system namely operator who can use our portal's "money deposit service" so that our portal can know which company deposited how much money to our portal's account in the bank.

1.4. TOOLS WE ARE USING

We have waited so much for getting Web Methods and Application Designer and when we get them we faced so many problems with installing those tools. Software AG were interested with that problems and arranged meeting for using tools twice, however, we had already started to learn building Web services on Tomcat 6.0 using Eclipse with Apache Axis2 plug-in at that time. The following is the list tools we are using.

- ECLIPSE with Apache Axis2 plug-in for development of POJO Web Services
- NETBEANS for development of Client of corresponding Web services (development of service consumers) using JAX-WS
- NETBEANS for development of JSP & JSF (Java Server Pages & Java Server Faces) pages of User Interface

- APACHE TOMCAT 6.0 & axis2 for deploying Java Server Pages, clients of Web services (i.e., Portal and other Job Seeking Web sites we are simulating)& Web Services.

- ESB for integration of services.

- MD5 for security

- PostgreSQL 8.0 as Database Management System.

ECLIPSE with Axis2 plug-in

Eclipse IDE with Axis2 plug-in provides an easy way of development of Web Services. We are developing our Web services using POJO method, and when we deploy our Web service in Apache Axis2 on Tomcat, the service methods became available to outside world.

- POJO (Plain old Java Object) Web services with Axis2: This a way of implementing Web services, a Web service development pattern by identifying the Java class of the service.

NETBEANS 5.5.1

NetBeans provide a very easy way of composing client side of Web services. All you have to do is to give the WSDL of the Web service you will use, or give its URL to the project. This is just like importing a package to the project. After you show the WSDL of a Web service, the stub code of the service is automatically created and you can use its available service methods just like using a method of an imported Java class.

- Stub code of a service: By using the WSDL of the available Web service, NetBeans creates a java class which provides us an easy way of using the methods of the service provider.

NETBEANS 5.5.1 Visual Web Pack: Visual Web Pack which is a patch for NetBeans 5.5.1 provides a very easy way of developing web application with user interface. Best part of creating user interface with a

JAVA IDE Platform such as NetBeans is you can specify what will be done, or what method will be called in source code of a component by the time you drag and drop it to the visual form page.

APACHE TOMCAT 6.0 & axis2

Apache Axis2 is a Web services engine and supports SOAP 1.1 and SOAP 1.2 and it also supports REST style of Web services. Up to now, we have used SOAP 1.1, and will most probably continue to use it, because we have written several Web services and their client for learning and trying some issues and we have not face any problem with messaging.

Axis2 is also efficient, modular and XML-oriented architecture. It supports easy additions of plug-in modules that provide additional features for WS-* standards such as security and reliability. WS-Addressing module is a part of Axis2 and the following modules are some example plug-in modules.

- Apache Sandesha2 module for WS-ReliableMessaging
- Apache Kandula2 module for WS-Coordination and WS-AtomicTransaction
- Apache Rampart for WS-Security

Reliable messaging, atomic transaction of messages and security of the messages are among the top most important considerations of SOA architecture. These tools will provide us to catch these WS-* standards in our project.

APACHE WSO2 as Enterprise Service Bus (ESB)

In our Project, we plan to use WSO2 Enterprise Service Bus as it supports the needed transformation, connectivity, mediation and management of web services. It is based on Apache Axis2 and Apache Synapse. As we use Apache Axis2 to deploy our services, WSO2 suits our system very well. On the other hand, WSO2 is known as XML and Web services centric Enterprise Service Bus.

The important point is how we will adapt WSO2 into our project. WSO2 ESB acts as the intermediary between client end services. The programs that have to be run are these: the destination server that hosts the ultimate service that has to be invoked to serve the client, the client itself, and the WSO2 ESB, which sits in between to perform the mediation. The destination server will be the server that we will put our Web services. WSO2 provides us the easily usable interface that makes easy

to follow the message mediation, because otherwise following the steps between client and server will be complicated. In addition to that, it can be seen the runtime statistics during message mediation. We know how we can use WSO2 ESB with our system but as our web service implementation has not been completed we could not implement this into our system. After completing the implementation of some of the web services, we will try to adapt WSO2 to our system.

MD5

MD5 is mostly used cryptographic technique which can take infinitely long data and convert it to 128 bits data. By doing this encryption, it separates data to 512 bits blocks and does the same operations to encrypt all of these blocks. All of the MD5 hashes will contain 32 characters. This means when you hash your 30 digits password it will be 32 digits and it will be also 32 digits when you hash 5 digits password.

MD5 is mostly used at registration systems, portals and the other sites that need membership. The process that is followed when you create your password is that:

Passwords are stored at database with its MD5 state when you first login to system. For the next time that you login the system, the system applies hash function on your password then search in database for this hashed value. As there is one way algorithm to encrypt when you forget your password, it cannot be accessed to the password instead it is reset and created again.

In our portal, we will use this cryptographic technique to encrypt the password that members use. On the other hand, to prevent all the risks, our SOAP messages is reached by someone else, we will use this technique in our portal to encrypt the values inside the tags of the XML in SOAP messages.

PostgreSQL 8.0 Database Management System

PostgreSQL is a very widely used database management system, besides it is open source, it performance is considerably good. We seesaw between MySQL and PostgreSQL at the beginning of the semester. Although all of the group members were experienced with MySQL, after reading comments about PostgreSQL, we decided to choose this option. We read that PostgreSQL is generally optimized and faster than other DBMSs for scenarios involving high transactional loads, high numbers of users (especially non-read-only applications), and complex queries, which is exactly what we need.

PART II - DATA DESIGN & DATA DICTIONARY

2.1. Data Objects of our Portal

Tables in our Portal's DB - JobSeekerPart

JobSeekerMembershipInfo

- Username
- Password
- SecretQuestion
- Answer
- Email

Addresses

- AddrId
- AddressAsText
- PostalCode
- Town
- City
- Country

CVs

- CVId
- CVName
- CVLanguage
- IsActive
- JobExperience
- CurrentWorkStatus
- additionalCVInfoAsText
- militaryServiceStatus
- Delay_Discharge_Date
- militaryExcuseCause
- drivingLicence1Class
- drivingLicence1IssueDate
- drivingLicence2Class
- drivingLicence2IssueDate
- TCNo
- Nationality
- Name
- Surname
- Gender
- BirthPlace_Country
- BirthPlace_City
- BirthDate
- MaritalStatus
- BloodGroup
- SmokingHabit
- AssociatedClubs
- Hobbies

References

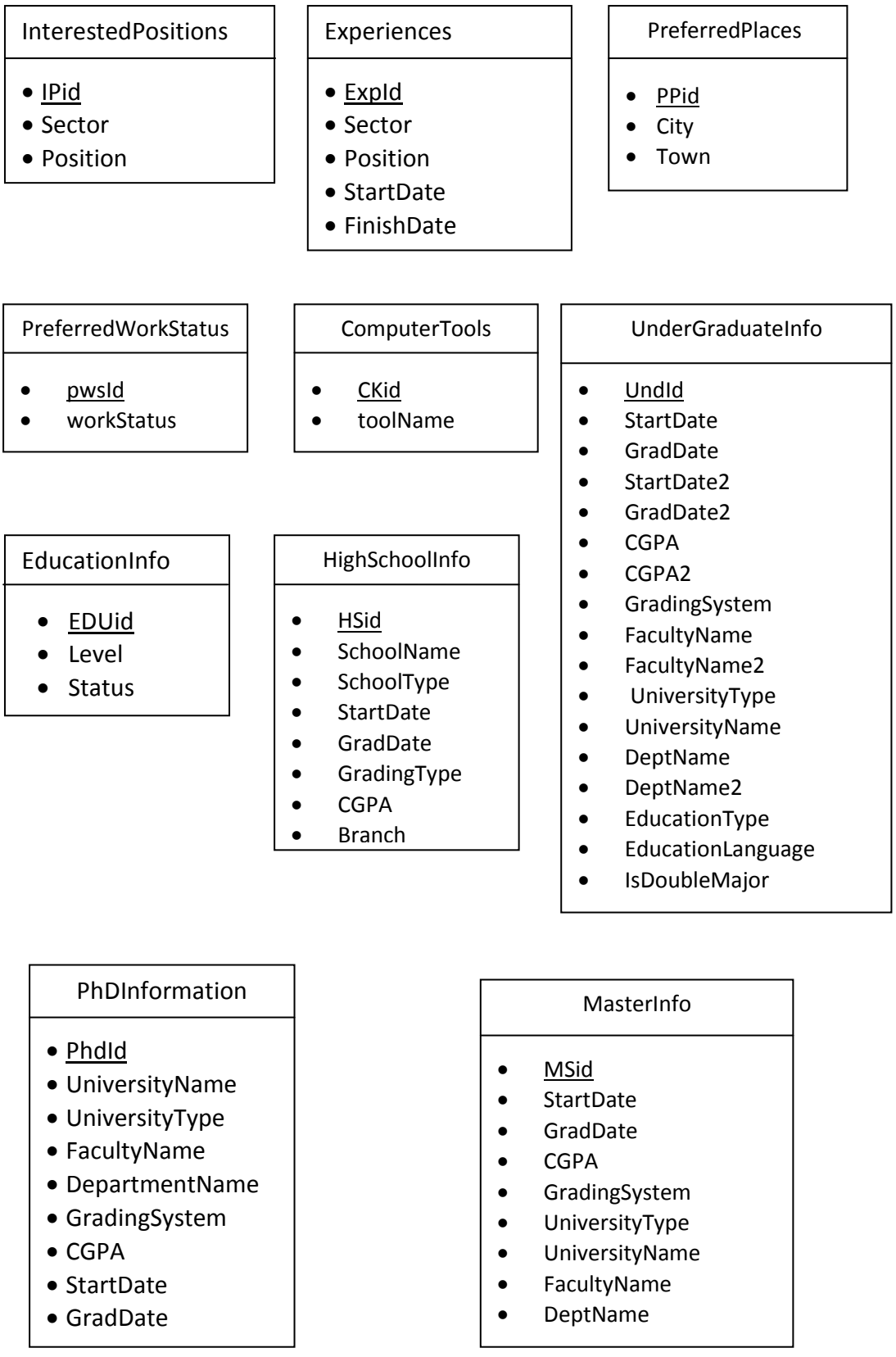
- RefId
- Name
- Surname
- FoundationName
- JobStatus

CommunicationInfo

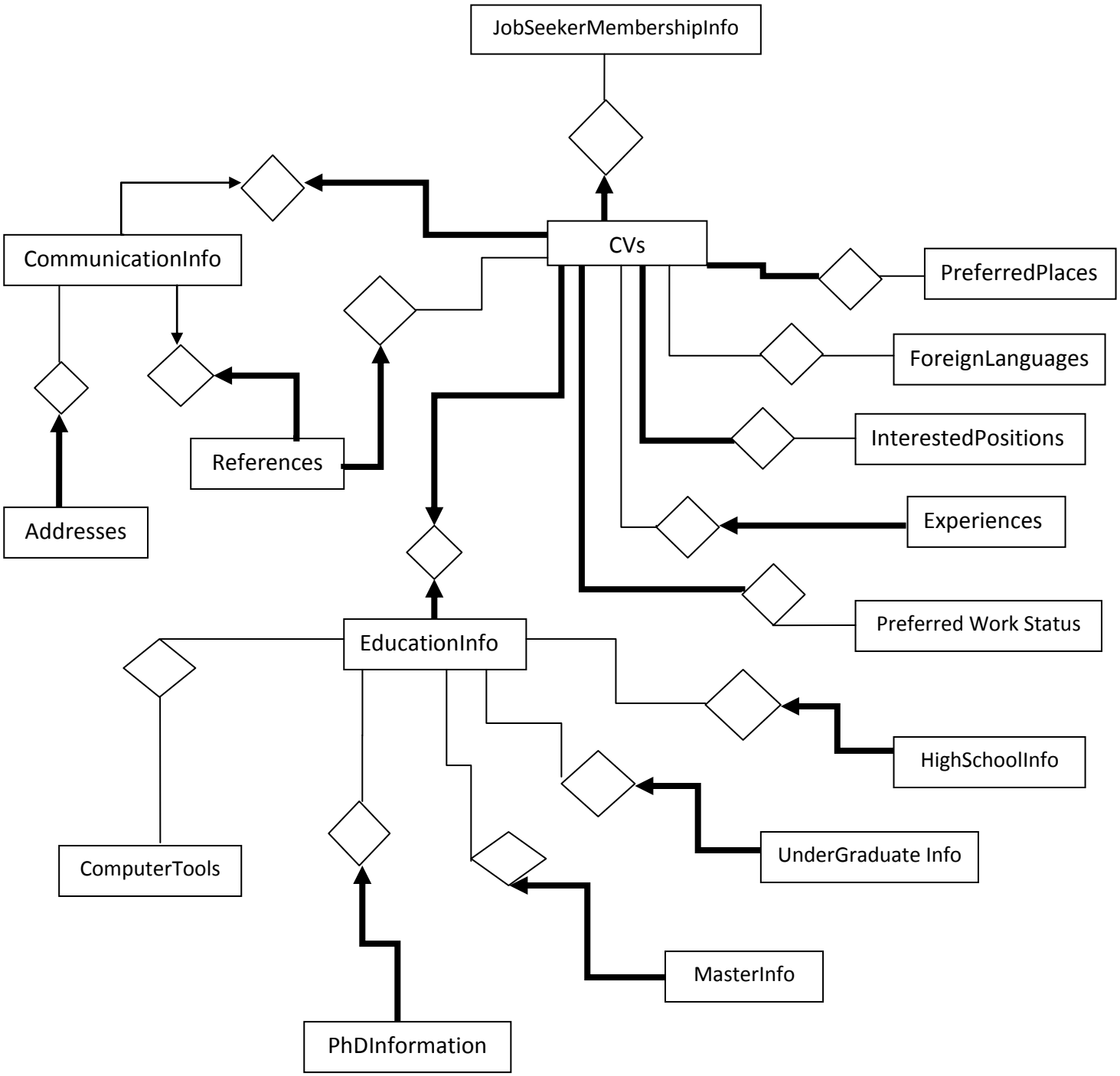
- ComId
- Email1
- Email2
- PreferredCommChannel
- CellPhone1
- CellPhone2
- HomeTel
- WorkPlaceTel
- WebPageURL

ForeignLanguages

- langId
- LanguageName


InterestedPositions
IPid
Sector
Position
Experiences
ExpId
Sector
Position
StartDate
FinishDate
PreferredPlaces
PPid
City
Town
PreferredWorkStatus
pwsId
workStatus
ComputerTools
CKid
toolName
UnderGraduateInfo
UndId
StartDate
GradDate
StartDate2
GradDate2
CGPA
CGPA2
GradingSystem
FacultyName
FacultyName2
UniversityType
UniversityName
DeptName
DeptName2
EducationType
EducationLanguage
IsDoubleMajor
EducationInfo
EDUid
Level
Status
HighSchoolInfo
HSid
SchoolName
SchoolType
StartDate
GradDate
GradingType
CGPA
Branch
PhDInformation
PhdId
UniversityName
UniversityType
FacultyName
DepartmentName
GradingSystem
CGPA
StartDate
GradDate
MasterInfo
MSid
StartDate
GradDate
CGPA
GradingSystem
UniversityType
UniversityName
FacultyName
DeptName

Complete ER Diagram of our Portal - Job SeekerPart



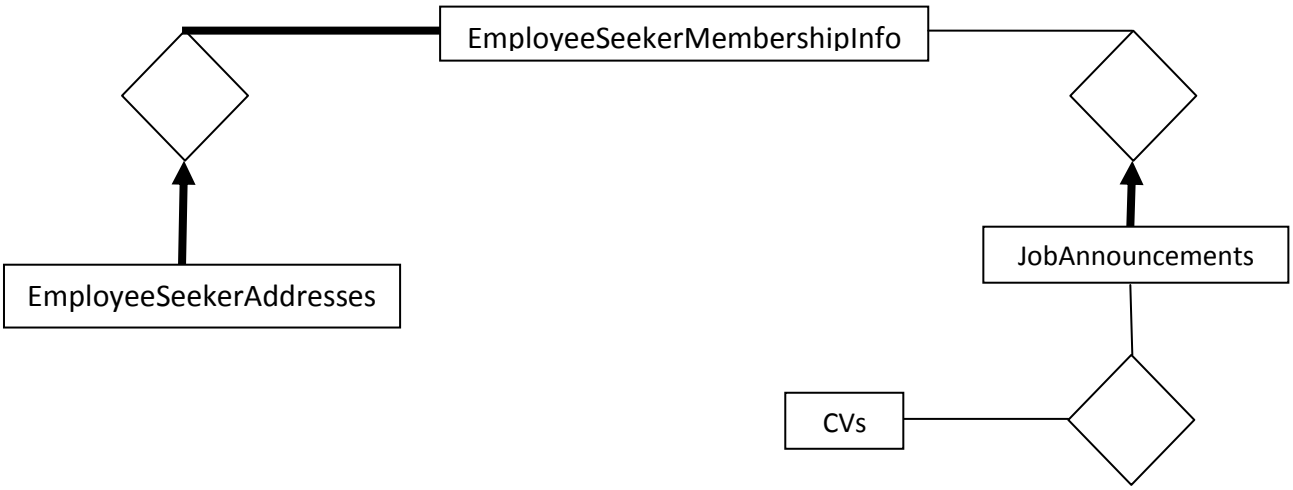
Tables in Our Portal's DB - Employee Seeker Part

EmployeeSeekerMembershipInfo
• userName • password • CompanyName • Sector • EmployeeNumber • WebPage • CashInTheBox • ResponsiblePersonName • ResponsiblePersonSurname • ResponsiblePersonEmail • StatusInTheCompany • Telephone • SecretQuestion • Answer

JobAnnouncements
• JAId • jobAnnouncementName • PositionInTheCompany • minExperienceYear • gender • militaryServiceRequirement • drivingLicenceRequirement • foreignLanguageRequirement • smokingHabitRequirement • minEducationLevelRequirement • acceptableEducationStatus • additionalJobInformationAsText • workPlaceCountry • workPlaceCity • workPlaceTown

EmployeeSeekerAddresses
• AddrId2 • AddressAsText • PostalCode • Town • City • Country • Tel1 • Tel2

Complete ER Diagram of our Portal - Employee SeekerPart



Tables in our Portal's DB - Remaining Part

JSWSs
• jswsId • jswsName • deptAmount

JobCVMatchArchive
• JCMId • companyName • companySector • jobPosition • educationLevelOfJobSeeker • educationStatusOfJobSeeker • approvalDate • workPlaceCountry • workPlaceCity • workPlaceTown

CV_JSWS_Queue
• CVid • DestinationJSWSid

EmployeeMembership_JSWS_Queue
• EmployeeUserName • DestinationJSWSid

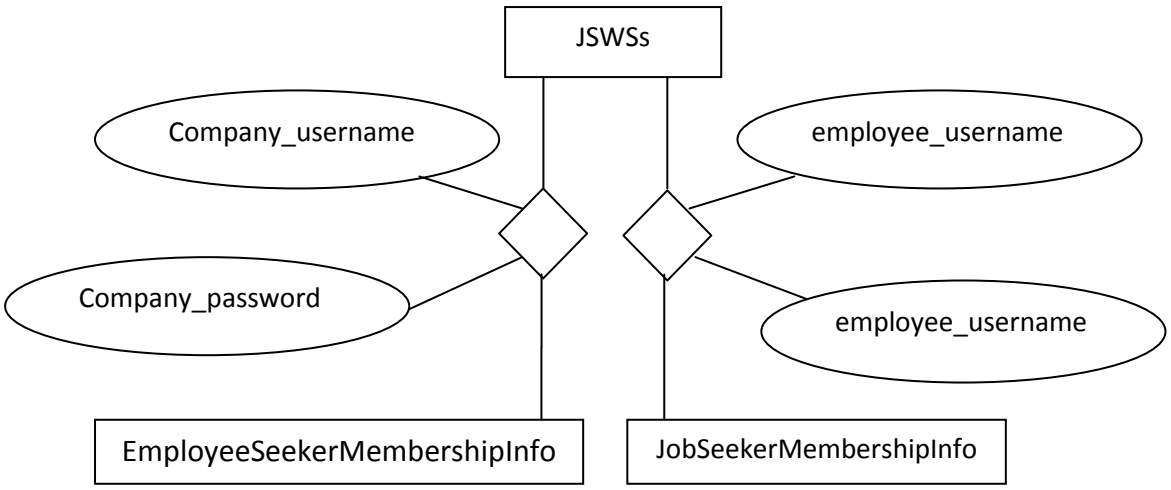
JobAnn_JSWS_Queue
• JAId • DestinationJSWSid

CompanyMembership_JSWS_Queue
• CompanyUserName • DestinationJSWSid

Updated_CV_JSWS_Queue
• CVownerUsername • CVNameBeforeUpdate • DestinationJSWSid

Updated_ JobSeekerMembership _Queue
• EmployeeUserName • DestinationJSWSid

Updated_ CompanyMembership _Queue
• CompanyUserName • DestinationJSWSid



Description of tables in our Portal's DB

1) JobSeekerMembershipInfo:

This table stores the membership information of job seekers. When a new user sign up to our portal, we directly write his/her membership information to our portal's database, and also add two records to EmployeeMembership_JSWS_Queue table because there are two JSWS websites under our portal and if there is a connection problem between those web sites and our portal, the membership information may be unable to send to those websites, so we periodically control that queue table to be able to send the unsent membership information of our portal users to JSWSs.

2) 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. Some of the CV object attributes are described below:

- **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, English or Turkish.
- **Delay_Discharge_Date**: According to the military service status, this field is used for storing the date of delay of or discharge from military service.
- **militaryExcuseCause** : If a job seeker don't do military service, the reason of this excuse must be specified. The possible values of this field are "Ogrenim Gormekte", "Cinsiyet Nedeniyle", "Hastalık Nedeniyle", "Diger Sebepler".

3) PreferredPlaces:

This table contains static information such that all cities and those cities' all towns in Turkey will be in this table. When a portal user wants to create a CV, he/she will choose from that places where he/she wants to work.

4) CommunicationInfo:

This table stores the communication information of job seekers and references.

5) ForeignLanguages:

This table contains static information such that all foreign languages, namely, English, French, German, ... A job seeker can select all languages he/she knows from this table by giving his/her speaking, writing and listening levels. Those selected languages will be added to his/her CV, there is no upper limit on this.

6) InterestedPositions:

This table also contains static information about all business sectors and positions. From this table, a job seeker can add the sectors and positions that he/she interested to his/her CV.

7) References:

When a job seeker creates a CV, he/she can add reference people and their communication information to his/her CV because Employers may want to communicate with those people before hiring the employee.

8) Addresses:

Addresses of job seekers and reference people will be stored in this table.

9) Experiences:

A job seeker can add his/her experiences to his/her CV in which the sector, position and date information that he/she worked take places.

10) Preferred Work Status:

This table stores static information about work status that a job seeker can accept. The possible values which can be stored in 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. A job seeker must select at least one work status. There will be a relationship between “Preferred Work Status” and “CVs”, and the information in the data table is stored into this relationship.

11) EducationInfo:

The possible values of the “Level” field are “İlköğretim Okulu”, “Lise”, “Üniversite”, “Master” or, “Doktora”. On the other hand, “Status” field can take the following values: “Mezun”, “Öğrenci” or “Terk”. These two fields will be used when we try to match CVs with unmatched job announcements.

12) HighSchoolInfo:

In this table, some of the possible values which can be stored in the “SchoolType“ field 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 seeker was. 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 so we will not explain them.

13) UnderGraduate Info:

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. This table’s “EducationType” field’s possible values are “Örgün Öğretim”, “İkinci Öğretim”, “Açık Öğretim” and “Uzaktan Öğretim”. “EducationLanguage” field is for specifying the education language of the university, for example in METU, it is English. “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 job seeker to the university. “isDoubleMajor” field will be true if a job seeker is/was registered to a double major program in his/her university. In that case, “FacultyName2”, “DeptName2”, “StartDate2”, and “CGPA2” fields are used for storing the information about the job seeker’s second major program. 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.

14) MasterInfo:

If a job seeker has a Master’s degree, he/she will fill the necessary fields in the “Create CV” form, and the information in those fields will be stored into this table.

15) PhDInformation:

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 into this table.

16) ComputerTools:

This table stores static information about computer tools, programs and programming languages which a job seeker can know how to use. Some of the values in this table are “Photoshop”, “Word”, “Excel”, “Powerpoint”, “Flash”, “C++”, “Java” 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.

17) EmployeeSeekerMembershipInfo:

When a company wants to become a member of our portal, we will store the information about this registration to this table. For every company, there must be a person who is responsible for “Telephone” field in this table belongs to the responsible person of the company.

18) EmployeeSeekerAddresses:

Since a company may have more than one address (because it can have many branches), we create an entity for the address information. In this table, there is also telephone information of the company. 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.

19) JobAnnouncements:

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.

20) JobCVMatchArchive:

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 the job announcement will be dropped from job seeking web site's database in which it was published and also from our portal's database and this CV-Job Announcement matching information will be stored to this table. The information in this table will be used when İş-Kur makes a query to our portal. Every night at 24:00, we will get that day's matches from JSWSs and add them to this table of our portal.

21) JSWSs:

This table stores all JSWSs under our portal. When a new JSWS applies to us to enter our JSWS-Portal system, we will add that JSWS name to this table of our portal. In the relation between this table

and EmployeeSeekerMembershipInfo table, we will store the information of companies' usernames and passwords in respective JSWSs. That is, when a company sign up to our portal, we will automatically make this company to be a member of all JSWSs under our portal, so those membership information must be stored somewhere in our portal's database and we will store them in the relation table specified previously. Likewise, we will have another relation table between this JSWSs table and JobSeekerMembershipInfo table and to this table, we will store the job seeker's membership information (username, password)in JSWSs which our portal create for them.

22) CV_JSWS_Queue:

When a job seeker creates a CV in our portal, we will directly store that CV to our portal's database and we add records to this CV_JSWS_Queue table about that CV whose meaning is that CV is not sent to any of the JSWSs under our portal. After that, another module which is triggered by a timer module or Send CV service module will fetch all records in this table, and send the unsent CVs to JSWSs.

23) JobAnn_JSWS_Queue:

When an employee seeker creates a job announcement and want to publish it in JSWSs through our portal, we will store this job announcement in our portal's database and will add this job announcement id to this queue table. This queue table, like the other queue tables in our database design, solves the problem of loosing data if a connection problem occurs between our portal and JSWSs. To make the point clear, assume a JSWS under our portal is not available when a job announcement is created and supposed to be published. If we didn't keep information about that a job announcement is not able to send to the destination JSWS, we would assume that job announcement is sent and published successfully, although it was not the case. So to solve this connection problem issue, we designed such queue tables which store the information of unsent things (In this case, job announcements).

24) CompanyMembership_JSWS_Queue:

This table has a similar service with the previous one, but this time unsent company membership information is stored.

25) EmployeeMembership_JSWS_Queue:

This table also has a similar service with the previous ones, but this time unsent job seeker membership information is stored.

26) updated_CV_JSWS_Queue:

When a job seeker updated his/her CV, this CV update must be reflected to JSWS databases. But to solve a possible connection problem between portal and JSWSs, we will add the reference of the updated CV and the JSWS's id whose database the change must be reflected to this queue table.

27) Updated_ JobSeekerMembership _Queue:

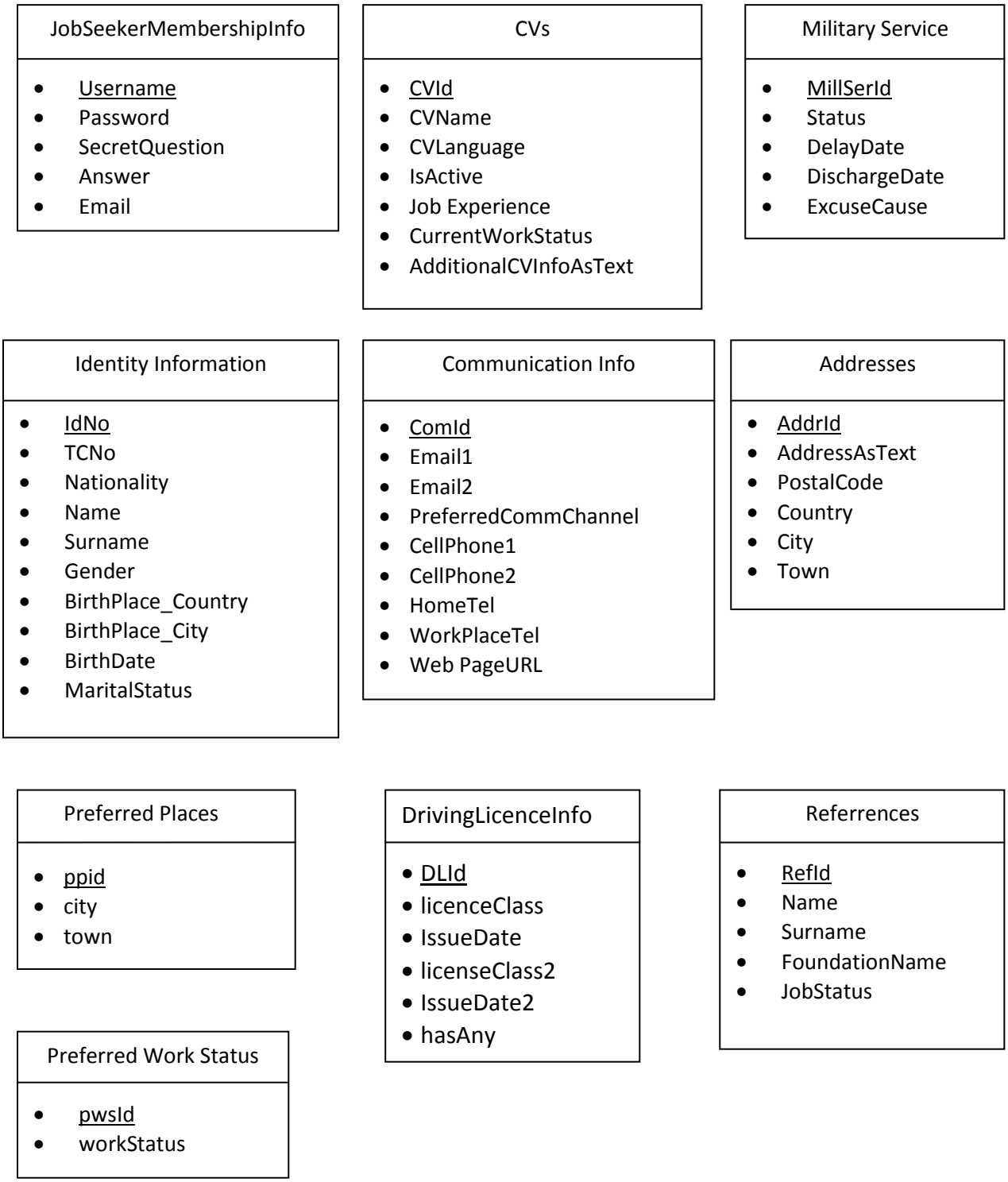
When a job seeker updated his/her membership, this update must be reflected to JSWS databases. But to solve a possible connection problem between portal and JSWSs, we will add the reference of the updated membership of the job seeker and the JSWS's id whose database the change must be reflected to this queue table.

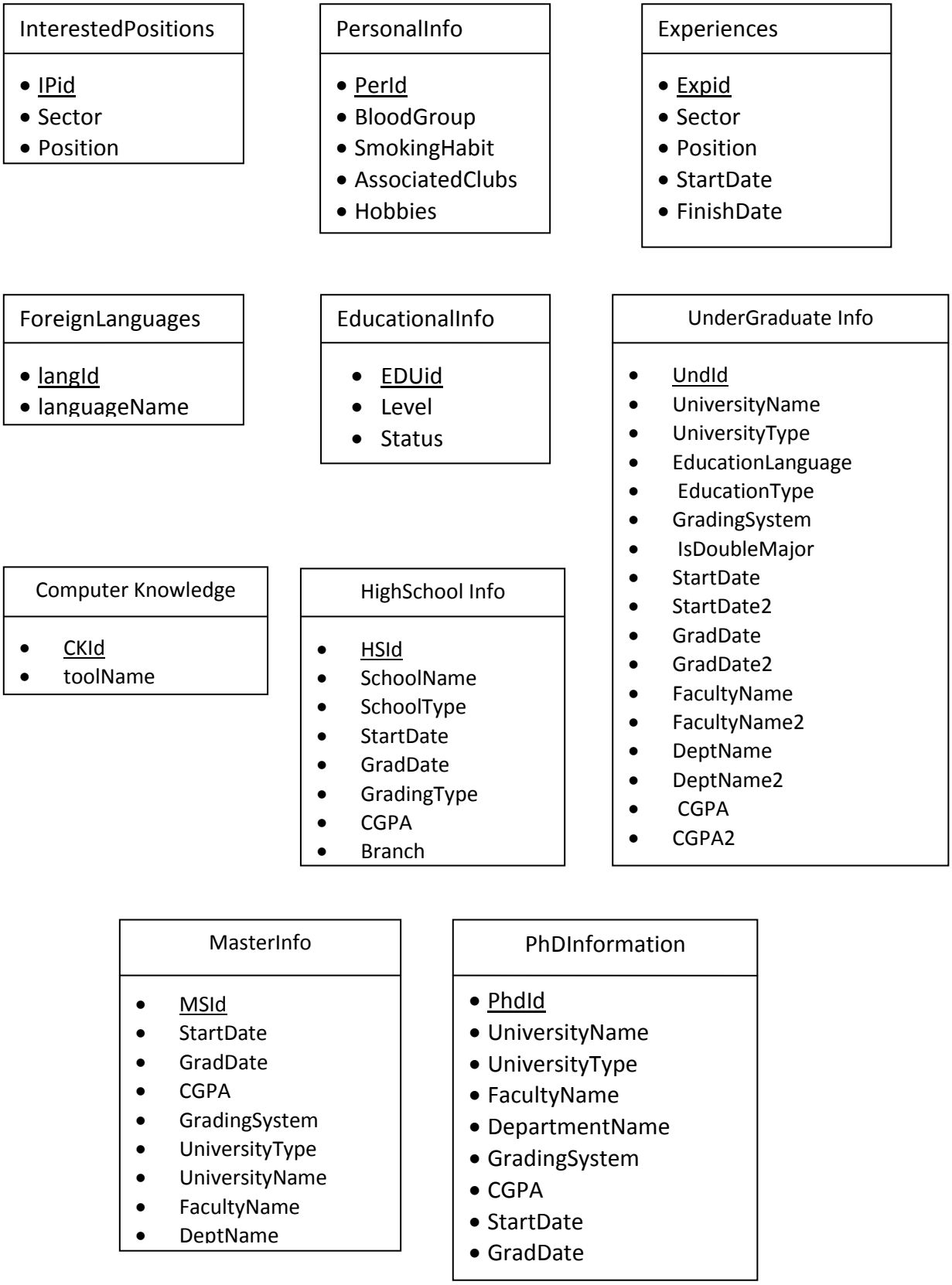
28) Updated_ CompanyMembership _Queue:

When a responsible person of a company updated his/her company's membership information, this update must be reflected to JSWS databases. But to solve a possible connection problem between portal and JSWSs, we will add the reference of the updated membership of the company and the JSWS's id whose database the change must be reflected to this queue table.

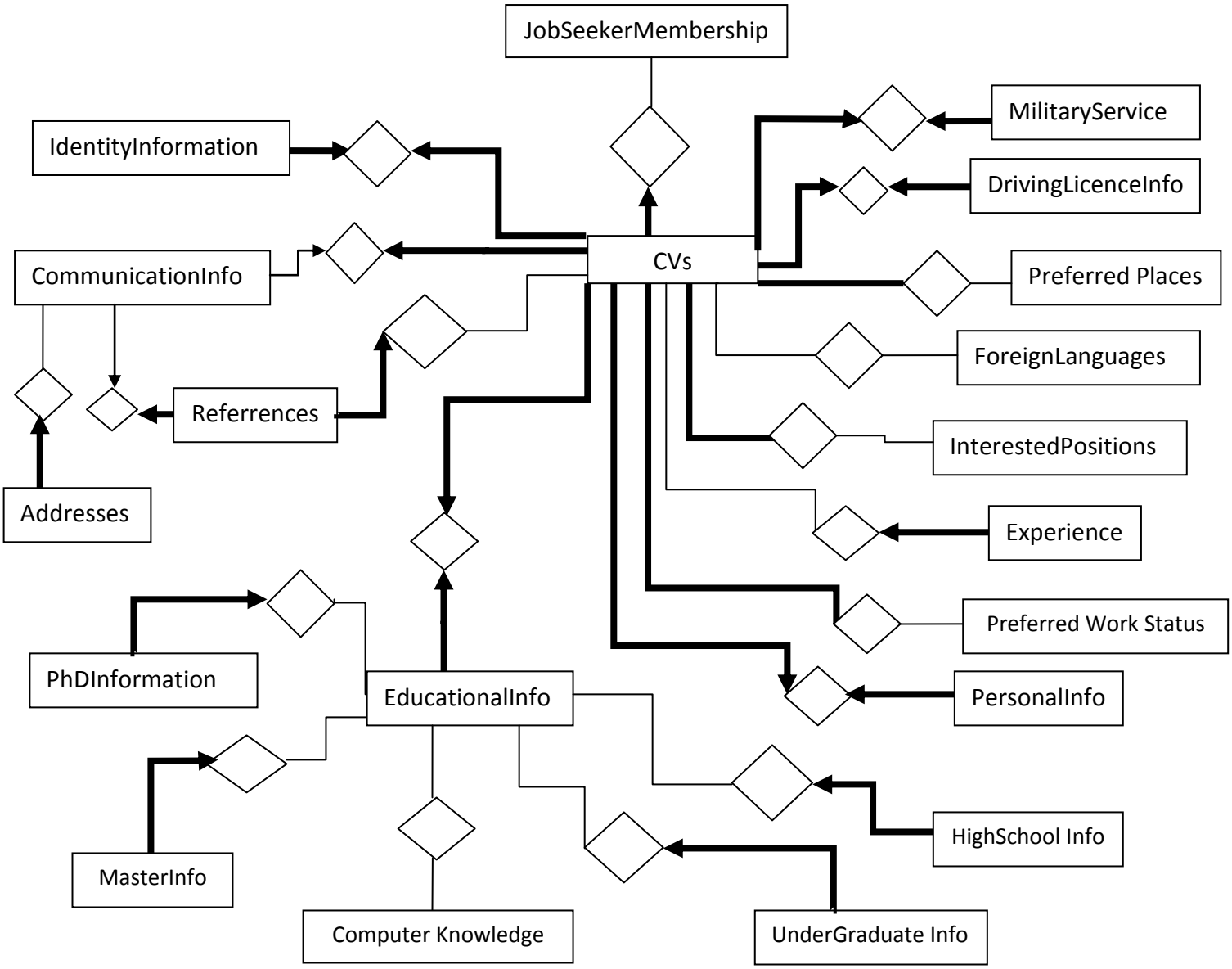
2.2. Data Objects of JSWS – 1 (Simulation of Kariyer.net)

Tables in JSWS1's DB – Job Seeker Part




InterestedPositions
IPid
Sector
Position
PersonalInfo
PerId
BloodGroup
SmokingHabit
AssociatedClubs
Hobbies
Experiences
Expid
Sector
Position
StartDate
FinishDate
ForeignLanguages
langId
languageName
EducationalInfo
EDUid
Level
Status
UnderGraduate Info
UndId
UniversityName
UniversityType
EducationLanguage
EducationType
GradingSystem
IsDoubleMajor
StartDate
StartDate2
GradDate
GradDate2
FacultyName
FacultyName2
DeptName
DeptName2
CGPA
CGPA2
Computer Knowledge
CKId
toolName
HighSchool Info
HSId
SchoolName
SchoolType
StartDate
GradDate
GradingType
CGPA
Branch
MasterInfo
MSId
StartDate
GradDate
CGPA
GradingSystem
UniversityType
UniversityName
FacultyName
DeptName
PhDInformation
PhdId
UniversityName
UniversityType
FacultyName
DepartmentName
GradingSystem
CGPA
StartDate
GradDate

Complete ER of JSWS 1 – Job Seeker Part



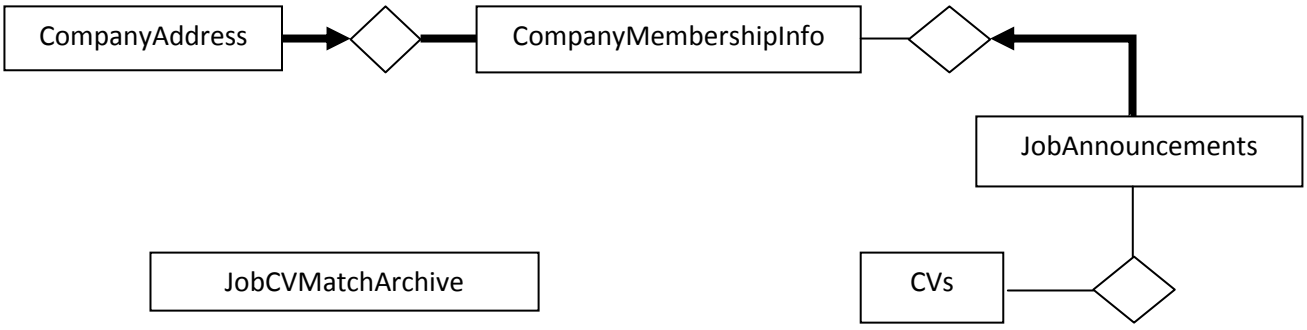
Tables in JSWS 1's DB – Employee Seeker Part

CompanyMembershipInfo
• userName • password • CompanyName • Sector • EmployeeNumber • WebPage • ResponsiblePersonName • ResponsiblePersonSurname • ResponsiblePersonEmail • StatusInTheCompany • Telephone • SecretQuestion • Answer

JobAnnouncements
• JAId • jobAnnouncementName • PositionInTheCompany • minExperienceYearRequirement • genderRequirement • militaryServiceRequirement • drivingLicenceRequirement • foreignLanguageRequirement • smokingHabitRequirement • minEducationLevelRequirement • acceptableEducationStatus • additionalJobInformationAsText • workplace_Country • workplace_City • workplace_Town

CompanyAddress
• addrID2 • Country • City • Town • postalCode • addressAsText • tel1 • tel2

Complete ER of JSWS 1 – Employee Seeker Part



Tables in JSWS 1's DB – Other Part

WebServiceUsers
• webserviceUser • receivableMoneyAmount

JobCVMatchArchive
• JCMId • companyName • companySector • jobPosition • educationLevelOfJobSeeker • educationStatusOfJobSeeker • approvalDate • workplace_Country • workplace_City • workplace_Town

Description of tables in our JSWS1's DB

1) JobSeekerMembershipInfo:

This table holds the job seekers membership information as the table name implies. We hold the unique username, password, secret question, answer to the secret question, and email information in this table. These are the basic information that a job seeker should have in order to have an account in Job Seeking Web Sites.

2) CVs:

This table holds one of the CVs of the user that are assigned to the related Job Seeking Web Site. The basic CV information like unique CV Id, name of the CV, language of the CV, the information about whether the CV is active or not, are hold together with some extra information about the CV owner.

3) PreferredPlaces:

This table holds the information about the desired work places of the job seeker. We give a unique id number for all of the city and town combinations of Turkey. When a user selects a prefered place from combo box in user interface, all we do is creating a connection between the id of the preferred place and the user's CV. The user can add as many place as he/she wants.

4) CommunicationInfo:

This table holds the communication information of a job seeker or a reference person of a job seeker. In this table, 2 email addresses, 2 cell phone numbers, 1 home telephone, 1 work telephone, the URL of web page of the user are stored. Since a job seeker may have more than one address, we separate "Addresses" table as a table showing this table.

5) ForeignLanguages:

This is a static table which holds the languages in the world with unique language id numbers. When a job seeker adds languages from combo box in the user interface, when he/she is creating a CV, he/she must also specify his/her speaking, writing and listening levels in that language. This level information will be stored in a relation table between this table and CV table.

6) InterestedPositions:

This table is also a static table like ForeignLanguages and PreferredPlaces tables. What we mean by static is that there are different sector positions combinations in this table. All these combinations have a unique id. When a job seeker adds sector and position preferences from the related combo box, we create a connection between the CV table and this one.

7) Referrences:

For holding the references of a job seeker, we have created the References table. We hold the name and surname of the reference person, foundation in which this person works and status of the person in that foundation.

8) Addresses:

In order to keep the address information of job seekers and their reference people, we designed this table. In this table, we keep the postal code, country, city and town fields to store the address information as a structured way. Apart from this, more specific but unstructured information is also kept as plain text in the “AddressAsText” field.

9) Experiences:

Although we hold the job experience of the job seeker in years as an attribute of the table “CVs”, we can also keep detailed information of the previous experiences of the job seeker by the means of this table. We hold the sector, position, start date, finish date of the previous experiences and give them a unique name “Expid”. A job seeker can add more than one experience information and in each declaration of an experience, a new row is added to this table.

10) Preferred Work Status:

This table is static and keeps information about all possible working status that a job seeker can accept to work. The possible working status are the following ones: “full time”, “half time”, “aggrement based”, “intern”, “project based” or “other”.

11) EducationalInfo:

The “Level” field in this table holds the information about the last school level of the job seeker in which he/she is/was student. Some of the possible values of this field are “high school”, “undergraduate”, “master”, etc. “Status” holds the information about the last situation of the job seeker about the last level he/she studied. The possible values are “still studying”, “graduated”, “left”. This entity is a middle entity between other education information tables and the CV.

12) HighSchoolInfo:

This table holds the high school information of job seekers. When a user tells that he is graduated from high school and enters information about that, a row is created in this table and connected to education information. We keep the school name, school type, start date, graduation date, grading type of the school, cummulative grand point average and the field of the school and a unique id number for representing this row of information. The field information of the school is holded by the “branch” field in the database. To give an example to this field definition, the job oriented high schools have different field like “engine”, “arts”, etc. “Branch” field holds this kind of information.

13) UnderGraduate Info:

In this table we keep the undergraduate school information of job seekers. We keep two group of information in this table: information about the university the member graduated and the status of the member at the time of graduation. You can see the details of this table from the ER diagram. We think the names are pretty clear. One thing we want to mention is why the information about member’s status is kept as double information. This is because that the member may have made double major. When he declares that the “IsDoubleMajor” becomes true. The second fields are filled according to the second major of the user.

14) MasterInfo:

If a job seeker has made a Master’s degree, this table will store the related information that job seeker gives from the user interface.

15) PhDInformation:

If a job seeker has made a Ph.D. degree, this table will store the related information that job seeker gives from the user interface.

16) Computer Knowledge:

We keep the computer tools, programs and programming languages that a job seeker is able to use in this table. This table will be a static table so there are predefined tools and programming languages like Microsoft Word, Macromedia Flash and C++. This table will be related with the "EducationInfo" table as can be seen from ER.

17) CompanyMembershipInfo:

This table is for keeping information about companies. A unique user name for the company is given to each company user. A matching password for company loginning is also kept in this table. For the cases the company forgets login information, we keep a secret question and answer. Since a company is not a real person, every company must have a responsible person. We keep the name, surname, email, status in the company and telephone of this person.

18) CompanyAddress:

This table is connected to the "CompanyMembershipInfo" table. Since a firm may have more than one address due to its branches, we designed this table instead of storing address information in company membership table.

19) JobAnnouncements:

This table holds the job announcements in the system. The defined job announcements are connected to the related company.

20) JobCVMatchArchive:

In this table, the job seeking web site keeps the information about approved CVs which were left to job applications published. Our portal will get that approval information dayly, and stores that information in portal's database to be able to generate the necessary statistics when "İş-Kur" makes query.

21) Personal Info:

This table keeps the social information of a member. These information are blood group, hobbies, smoking habit and the clubs that user is a member of.

22) Identity Information:

This table keeps most of the information you can find in a national identity card.

23) Military Service:

This information gives information about the military service information. The delay information or start – end date like information is kept in this table.

24) DrivingLicenceInfo:

The driving information is kept in this table. For every job seeker, at most two driving licence information can be stored.

25) WebServiceUsers:

Since this JSWS provides web services to our portal to publish a job announcement, for every job announcement JSWS records money information that our portal must give to it. This money information is also stored in portal's database. Periodically, as real money transactions occur between JSWS and our portal, the receivableMoneyAmount field will be updated accordingly.

2.3. Data Objects of JSWS – 2 (Simulation of yenibiris.com)

Tables in JSWS2's DB – Job Seeker Part

IsArayanUyelikBilgileri

- kullaniciAdi
- sifre
- gizliSoru
- cevap
- eposta

Ozgecmisler

- ozgecmisNo
- ozgecmisAdi
- ozgecmisDili
- aktifMi
- isTecrubesi
- simdikiCalismaDurumu
- ozgecmisAyrintisi
- kanGrubu
- sigaraAliskanligi
- uyeKlupler
- Hobiler
- AskerlikDurumu
- Tecil_Terhis_Tarihi
- MuafiyetNedeni

IletisimBilgileri

- iletisimNo
- eposta1
- eposta2
- tercihEdilenIletisimYolu
- cepTelefonu1
- cepTelefonu2
- evTelefonu
- isTelefonu
- websayfasiAdresi

KimlikBilgileri

- kimlikNo
- vatandaslikNo
- irk
- isim
- soyisim
- cinsiyet
- dogduguUlke
- dogduguSehir
- dogumTarihi
- medeniHali

Adresler

- adresNo
- acikAdres
- postaKodu
- Ulke
- Il
- Ilce

Referanslar

- referansNo
- Isim
- Soyisim
- calistigiKurumunAdi
- KurumdakiPozisyonu

TercihEdilenYerler

- yerNo
- il
- ilce

SurucuBelgesiBilgileri

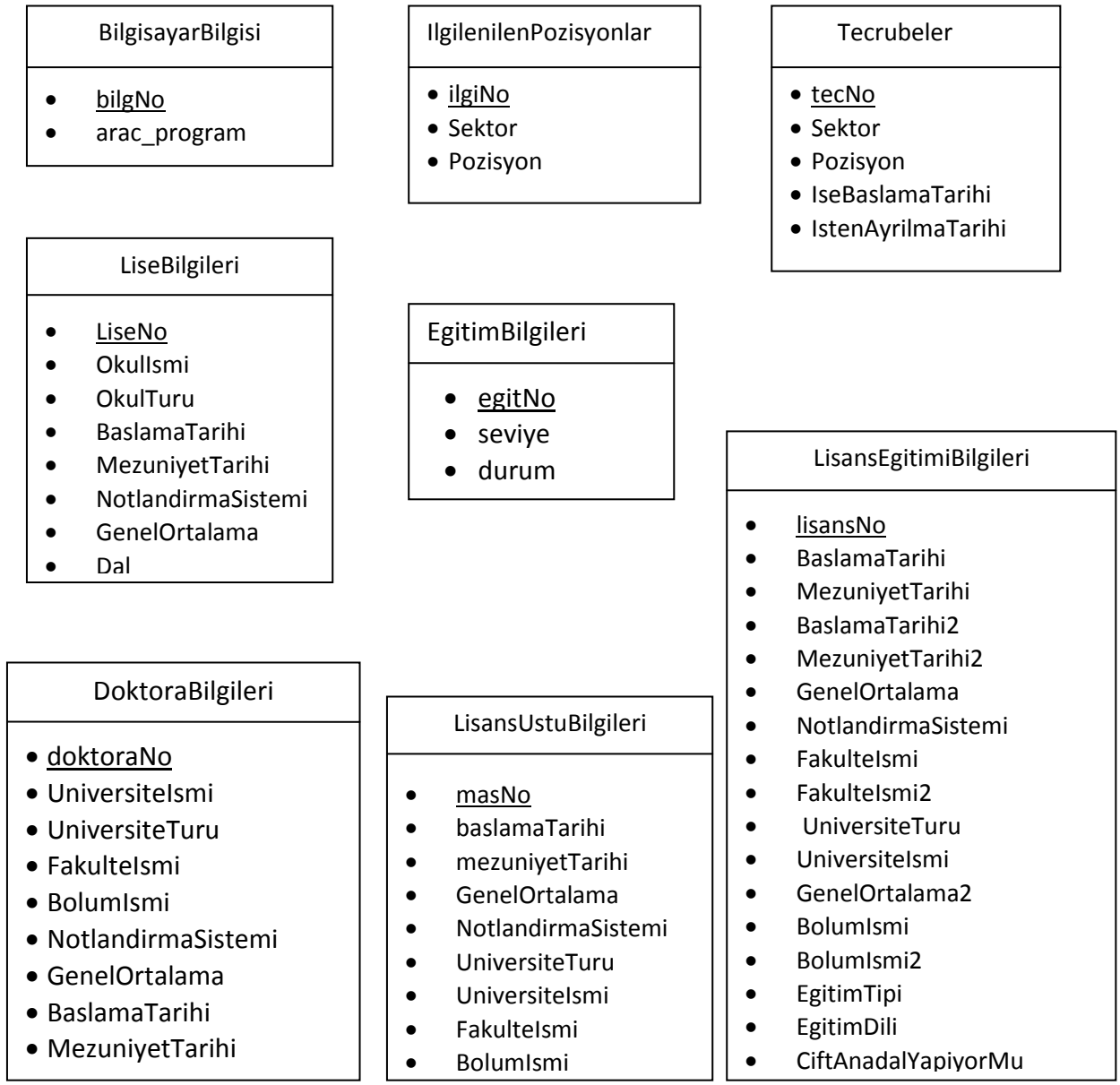
- surNo
- varMi
- sinifi
- verilisTarihi
- sinifi2
- verilisTarihi2

TercihEdilenCalismaSekli

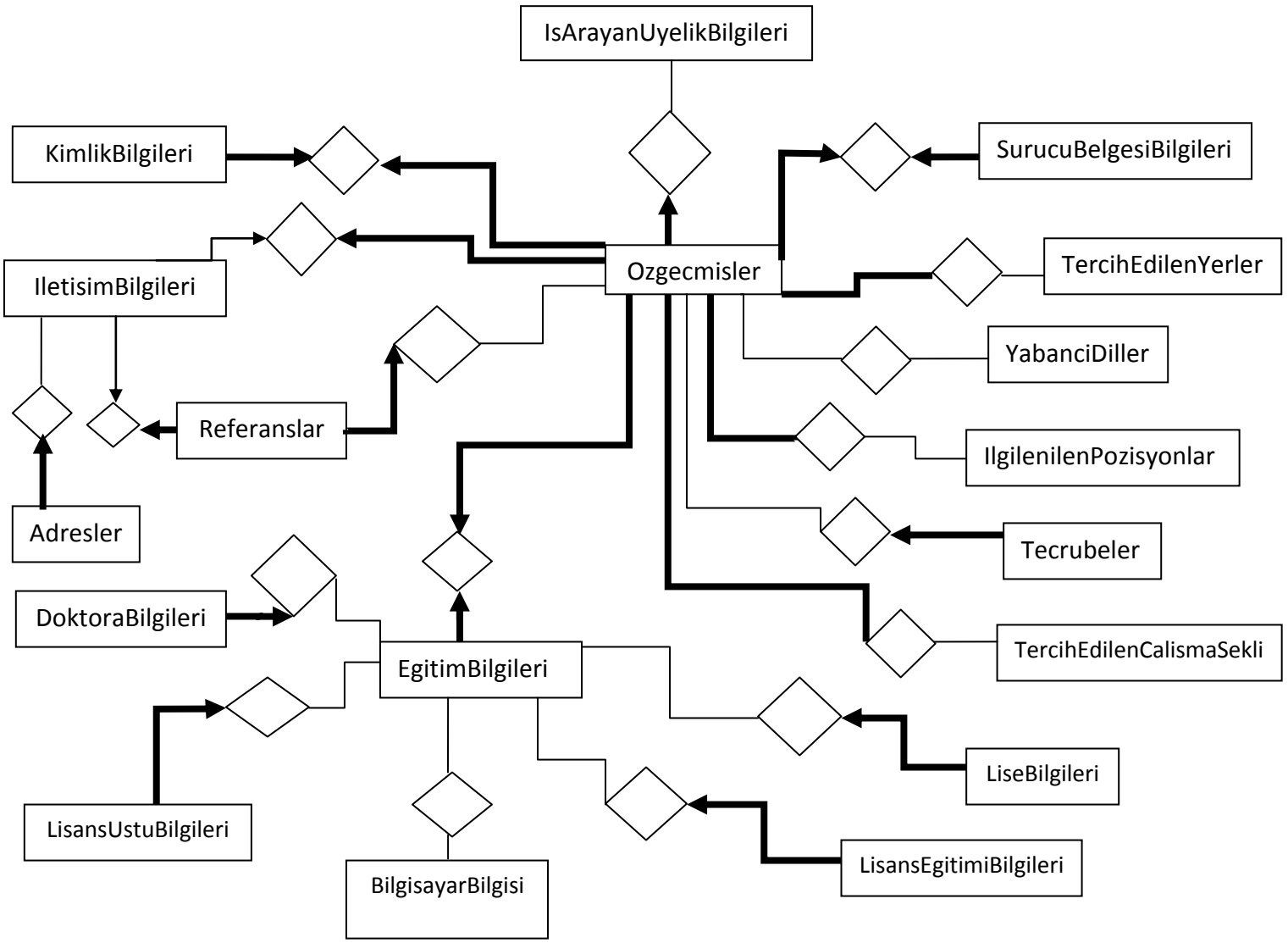
- calismaNo
- calismaSekli

YabanciDiller

- dilNo
- dilAdi


BilgisayarBilgisi
bilgNo
arac_program
IlgilenilenPozisyonlar
ilgiNo
Sektor
Pozisyon
Tecrubeler
tecNo
Sektor
Pozisyon
IseBaslamaTarihi
IstenAyrilmaTarihi
LiseBilgileri
LiseNo
OkulIsmi
OkulTuru
BaslamaTarihi
MezuniyetTarihi
NotlandirmaSistemi
GenelOrtalama
Dal
EgitimBilgileri
egitNo
seviye
durum
LisansEgitimiBilgileri
lisansNo
BaslamaTarihi
MezuniyetTarihi
BaslamaTarihi2
MezuniyetTarihi2
GenelOrtalama
NotlandirmaSistemi
FakulteIsmi
FakulteIsmi2
UniversiteTuru
UniversiteIsmi
GenelOrtalama2
BolumIsmi
BolumIsmi2
EgitimTipi
EgitimDili
CiftAnadalYapiyorMu
DoktoraBilgileri
doktoraNo
UniversiteIsmi
UniversiteTuru
FakulteIsmi
BolumIsmi
NotlandirmaSistemi
GenelOrtalama
BaslamaTarihi
MezuniyetTarihi
LisansUstuBilgileri
masNo
baslamaTarihi
mezuniyetTarihi
GenelOrtalama
NotlandirmaSistemi
UniversiteTuru
UniversiteIsmi
FakulteIsmi
BolumIsmi

Complete ER of JSWS 2 – Job Seeker Part



Tables in JSWS 2's DB – Employee Seeker Part

SirketUyelikBilgileri
• kullaniciAdi • sifre • gizliSoru • cevap • sirketIsmi • sirketSektoru • calisanSayisi • websayfasi • sorumluKisiAdi • sorumluKisiSoyadi • sorumluKisiEpostaAdresi • sirkettekiPozisyonu • telefonu

IsIlanlari
• ilanNo • ilanIsmi • sirkettekiPozisyon • minimumIsTecrubesi • cinsiyet • askerlikGereksinimi • surucuBelgesiGereksinimi • yabanciDilGereksinimi • sigaraAliskanligiGereksinimi • minimumEgitimSeviyesi • kabulEdilebilirEgitimDurumu • isIlaniAyrintisi • calismaYeri_ Ulke • calismaYeri _ Il • calismaYeri _ Ilce

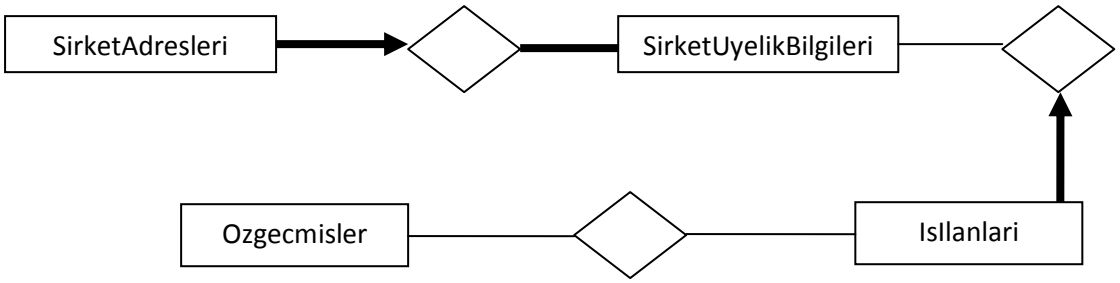
SirketAdresleri
• sirketAdresNo • Ulke • Il • Ilce • postaKodu • acikAdres • tel1 • tel2

Tables in JSWS 2's DB - Other Part

IseKabulArsivi
• arsivNo • sirketAdi • sirketSektoru • isPozisyonu • iscininEgitimSeviyesi • iscininEgitimDurumu • iseAlimTarihi • calismaYeri_Ulke • calismaYeri _Il • calismaYeri _Ilce

WebServisKullanicilari
• kullaniciIsmi • alacakMiktari

Complete ER of JSWS 2 – Employee Seeker Part



Description of tables in our JSWS 2's DB

In fact, this database is similar to the JSWS1's database but also has some differences. The first difference is this database belongs to a Turkish Job Seeking Web Site so the table names and and table field names are in Turkish. There are also some architectural differences. We will mention these differences while we are giving the definitions of our tables below.

1) IsArayanUyelikBilgileri:

This table is almost the same with the "JobSeekerMembershipInfo" table and holds the main membership information. The only difference is that the field names which are written in Turkish.

2) Ozgecmisler:

This table is like a combination of the tables "CVs", "PersonalInfo" and "Military Service" tables of JSWS. It doesn't match with Military Service table with a one to one relation. This is designed such that because we wanted to show there can be differences between job seeking web sites's database designs.

3) TercihEdilenYerler:

This table like in the "Preferred Places" table in JSWS1, holds the information about the desired work places of the job seeker.

4) IletisimBilgileri:

This table like in the “Communication Info” table in JSWS1, holds the communication information of a job seeker or a reference person of a job seeker.

5) YabanciDiller:

The aim of this table is the same with the table “ForeignLanguages”.

6) IlgilenilenPozisyonlar:

This table is the same with the table “InterestedPositions”. The information in that table is also valid for this table.

7) Referanslar:

This table is the same with the table “Referrences”. The information in that table is also valid for this table.

8) Adresler:

This table is the same with the table “Addresses”. The information in that table is also valid for this table.

9) Tecrubeler:

This table is the same with the table “Experiences”. The information in that table is also valid for this table.

10) TercihEdilenCalismaSekli:

This table is the same with the table “Preferred Work Status”. The information in that table is also valid for this table.

11) EgitimBilgileri:

This table is the same with the table “EducationalInfo”. The information in that table is also valid for this table.

12) LiseBilgileri:

This table is the same with the table “High School Info”. The information in that table is also valid for this table.

13) LisansEgitimiBilgileri:

This table is the same with the table “UndergraduateInfo”. The information in that table is also valid for this table.

14) LisansUstuBilgileri:

This table is the same with the table “MasterInfo”. The information in that table is also valid for this table.

15) DoktoraBilgileri:

This table is the same with the table “PhDInformation”. The information in that table is also valid for this table.

16) BilgisayarBilgisi:

This table is the same with the table “ComputerKnowledge”. The information in that table is also valid for this table.

17) SirketUyelikBilgileri:

This table is the same with the table “CompanyMembershipInfo”. The information in that table is also valid for this table.

18) SirketAdresleri:

This table is the same with the table “CompanyAddress”. The information in that table is also valid for this table.

19) IsIlanlari:

This table is the same with the table “JobAnnouncements”. The information in that table is also valid for this table.

20) IseKabulArsivi:

This table is the same with the table “JobCVMatchArchive”. The information in that table is also valid for this table.

21) KimlikBilgileri:

This table is the same with the table “Identity Information”. The information in that table is also valid for this table.

22) SurucuBelgesiBilgileri:

This table is the same with the table “DrivingLicenceInfo”. The information in that table is also valid for this table.

23) WebServisKullanicilari:

Since JSWS2 provides web services to our portal to publish a job announcement, for every job announcement JSWS2 records money information that our portal must give to it. This money information is also stored in portal's database. Periodically, as real money transactions occur between JSWS2 and our portal, the alacakMiktari field will be updated accordingly.

2.4 HOW WE ARE GOING TO SOLVE THE PROBLEM CREATED BY THE DIFFERENT RELATIONAL SCHEMAS ON DIFFERENT CAREER WEB SITES

As we showed in data objects part of the report, each Job seeking Web site has its' own database structure, own relational schemas. To solve this problem, we have to create different XML formatted messages for each career Web site from the same information provided by the Job seeker or Company. For example, a person wants to become a member of our portal and so all the career web sites working under agreement with us, and filled the registration form. The information taken from the user interface components is appropriate for our database structure, however, not appropriate for other career Web sites. They cannot take the XML message content of which is appropriate for our database relational schemas. Therefore, we have to parse that information taken from the user interface for each career Web site (i.e., Job Seeking Web Sites).

For that reason we have different "Message Composer" sub-modules within each client sending request message, for each JSWS. Each "Message Composer" sub-modules within a client, takes the same message and converts to appropriate form for corresponding JSWS.

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.

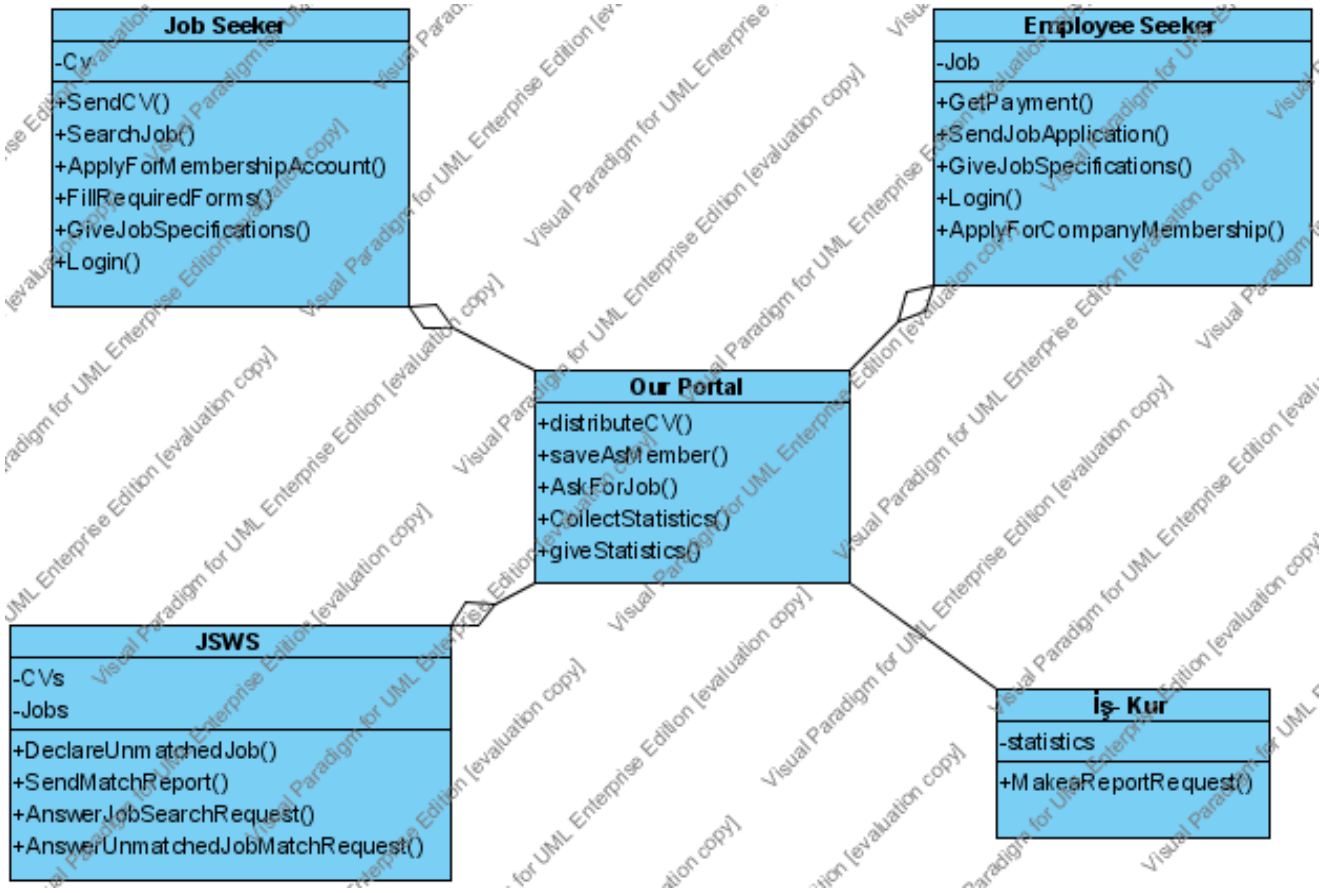
- The middle layer is the "**Application layer**" running on the server of our portal. There will be two different class of application servers (servlets) :

    - 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



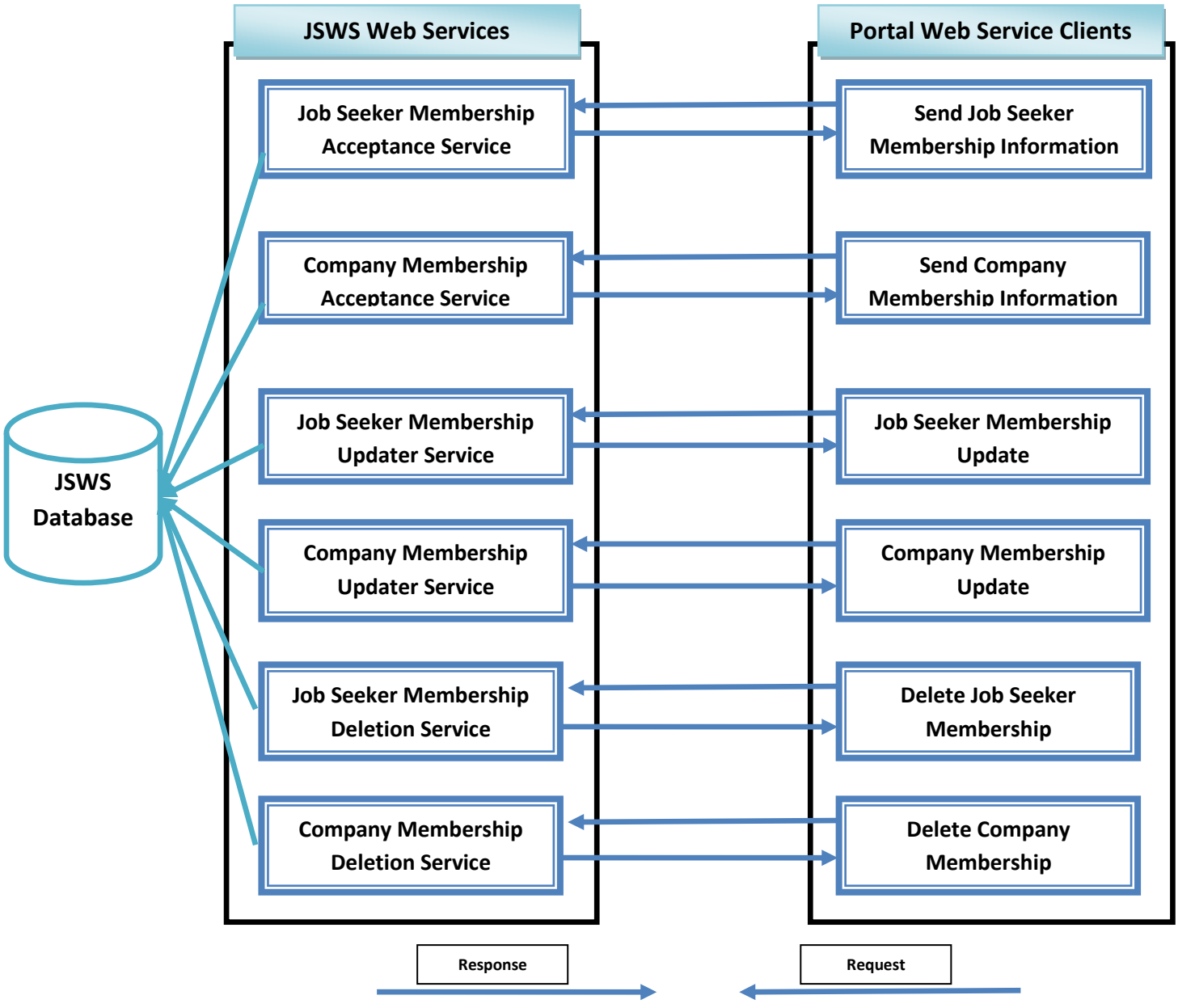
If we explain our class diagram, in the diagram there are five classes because there are 5 specific types.

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. DETAILED LOOK TO THE SERVICES DESIGN & FLOW OF DATA

Web services provide a standard way to implement a business function that can be invoked remotely. They support interoperability by separating the mechanisms of access from the implementation. All communication between our portal and the career web sites (the JSWSs) is done via calling remote procedures (RPC – remote procedure call) of the Web services. When we create a service for outside world, the clients can access the service's methods with information WSDL of the service provide. Now, let's look to the services of the Career Web sites.

Membership Related Services & Their Clients



Note that, we have shown just one JSWS in the figures because our aim was to show the inter relationships between service providers and the service consumers, and since the relationship between our portal and the JSWS web services are the same for each JSWS, we show one JSWS representing all.

The figure above is an abstract look to the membership related services. In this figure, our portal is in a position of service consumer in all situations. For example, let's take "Send Company Membership Information" service. This service is the client of the "Company Membership Acceptance Service", which is the "Service Provider" because our portal sends a request message that says: "Would you accept this Company as a member?", and the career web sites' service (Company Membership Acceptance Service) says "Yes, I would" or "No, I won't, this message is not a format as our database's".

Now let's take a deeper look to these service providers and service consumers inter relationship one by one.

When a person wants to become a user as a job seeker, he will fill the registration form. And "Database Utility" module will take that registration information from the textboxes, comboboxes or from the other user interface components and compose an SQL insert command for both "Job Seeker Membership" table and the "Unsent Membership information queue". That queue in fact, is just a table, with a set of rows having two information inside each:

- JSWSid
- Membership Id

From the JSWS id, the system understands to which Job Seeking Web Sites the membership information is unsent. And from the Membership Id, it understands which membership information is unsent. For example, assume there are five career Web sites working under agreement with us. Assume person wanted to become a member to all these Web sites; he filled the registration form of our portal. Database Utility module inserted that person's membership information to our database, and assume that our DBMS assign "135" as a membership id (By the way, that id is nothing to do with the person, it is just an information for us for accessing rows more easily). Then the Database Utility module insert five rows to our unsent Job seeker membership queue like:

- 1 135
- 2 135
- 3 135
- 4 135
- 5 135

These five rows means, the membership with the id of 135 is unsent to JSWS-1, JSWS-2, JSWS-3, JSWS-4, JSWS-5.

Also “Personal Membership Information Sending Service” is invoked as the user registers to our portal. With this trigger, Personal Membership Information Sending Service looks to the queue of this module, and takes the all membership information of each membership id included in the queue. And looks to which JSWSs these membership informations are unsent. After taking the membership information, it gives that information to the related JSWSs “Message Composer” modules. For each JSWS we have different “Message Composer” modules since their database structures are different from each other. Message Composers takes the information taken from the database, and converts that formatted information to appropriate XML messages.

Those XML messages are taken by related JSWS message sender modules and sent to the related JSWS as SOAP messages. In fact the exact client of the “Job Seeker Membership Acceptance Service” is these message senders. Because, these message senders invoke the remote procedures of the Web services on the JSWSs’ side.

Those XML messages are taken by “Job Seeker Membership Acceptance Service” and the information within the message is controlled by the “Job Seeker Membership Info Controller” and if there is an error in the format, Error message is returned to our portal by the JSWS response sender. Otherwise the information within the XML message is passed to the Database Utility module of the JSWS and that membership information is inserted to the Database of it. If an error condition occurs somehow, the message returned by the Database module is passed to the JSWS response sender. If no error condition occurs, JSWS response sender sends us a message meaning “membership is accepted and no error condition has occurred”.

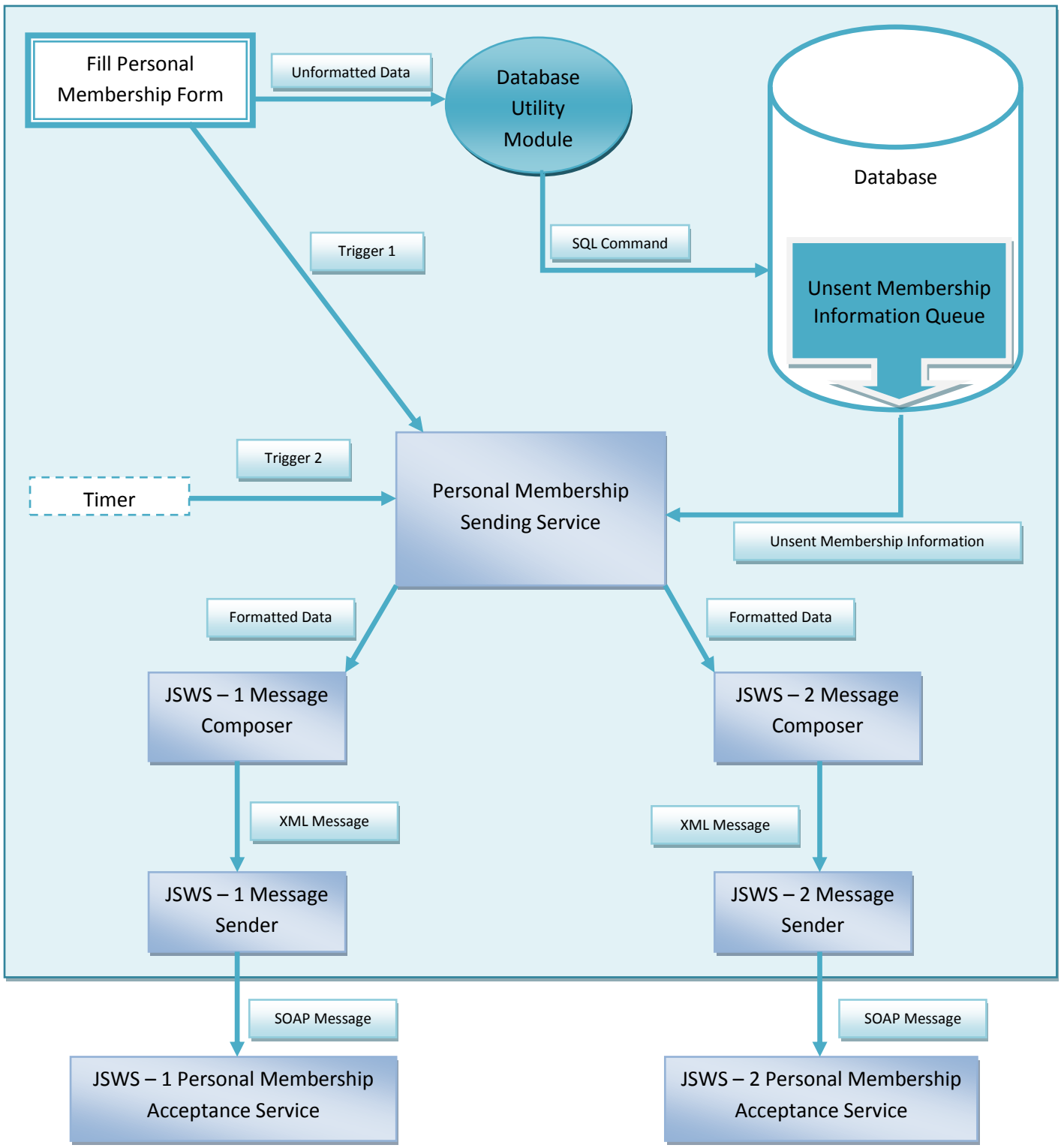
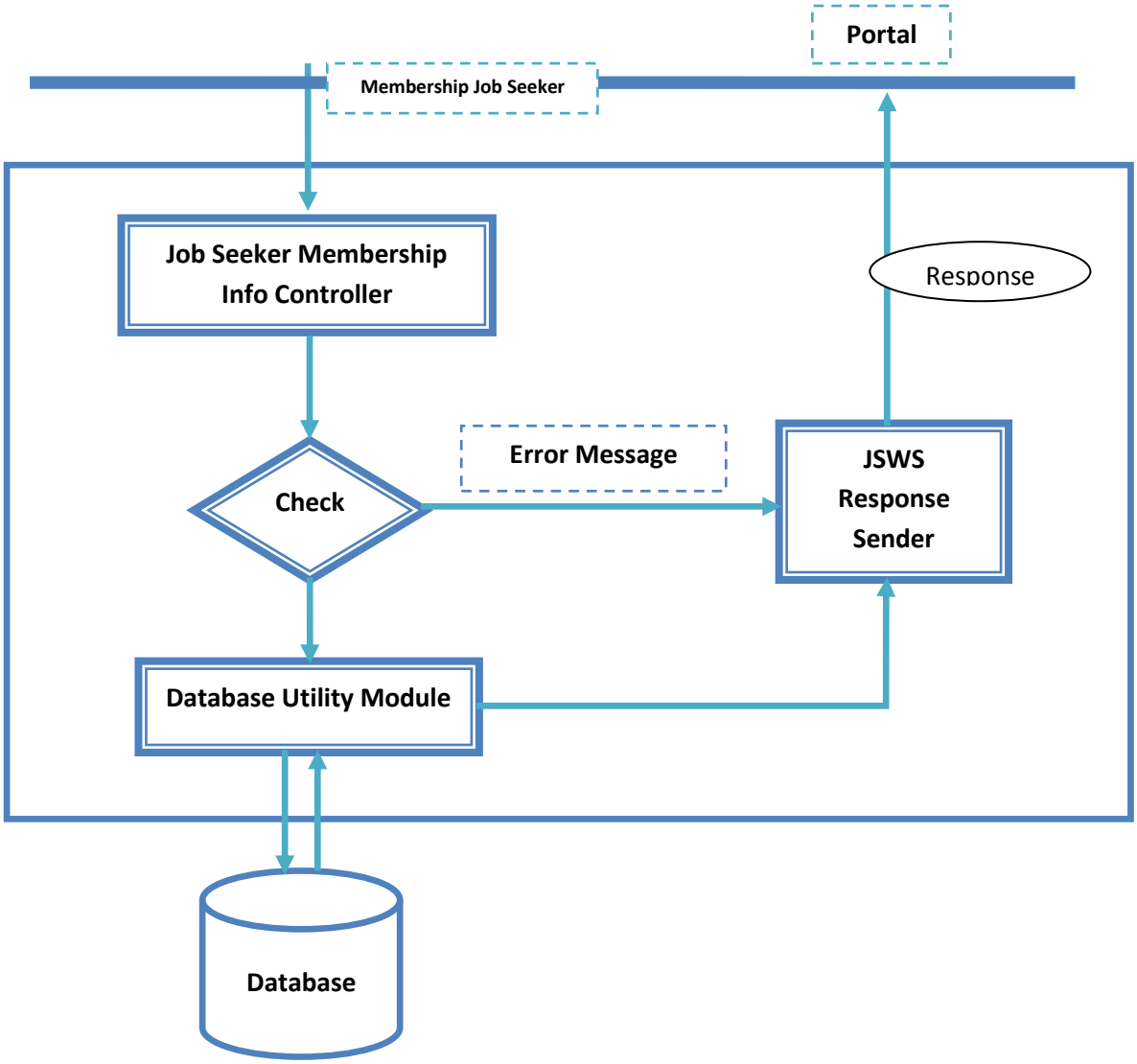


Figure of Job Seeker Memership Sending Module



Other Service Consumer – Service Provider relationships have a very similar process, there are just a little bit differences. So we need not to explain them, because they are almost same.

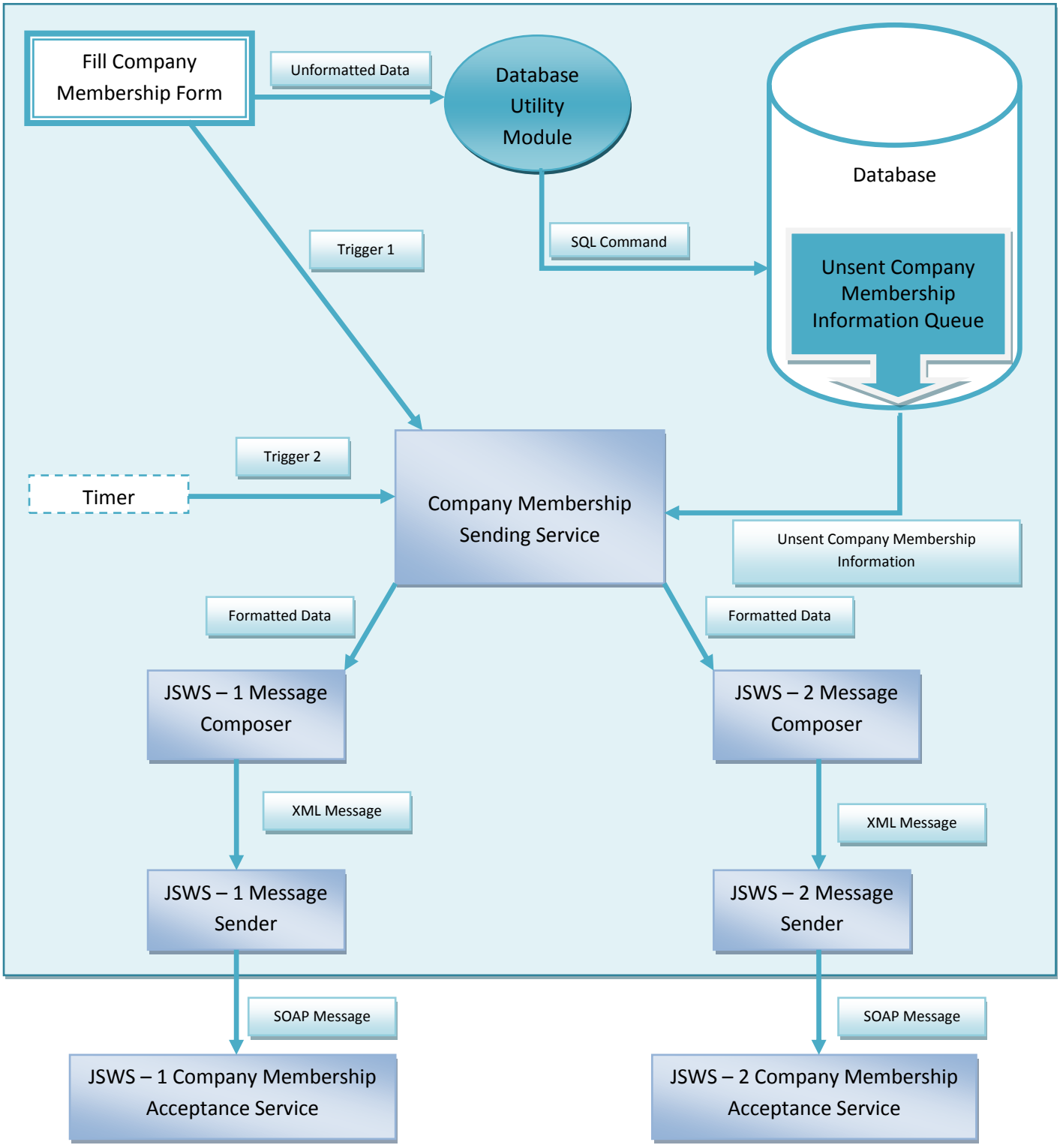


Figure of Company Memership Sending Module

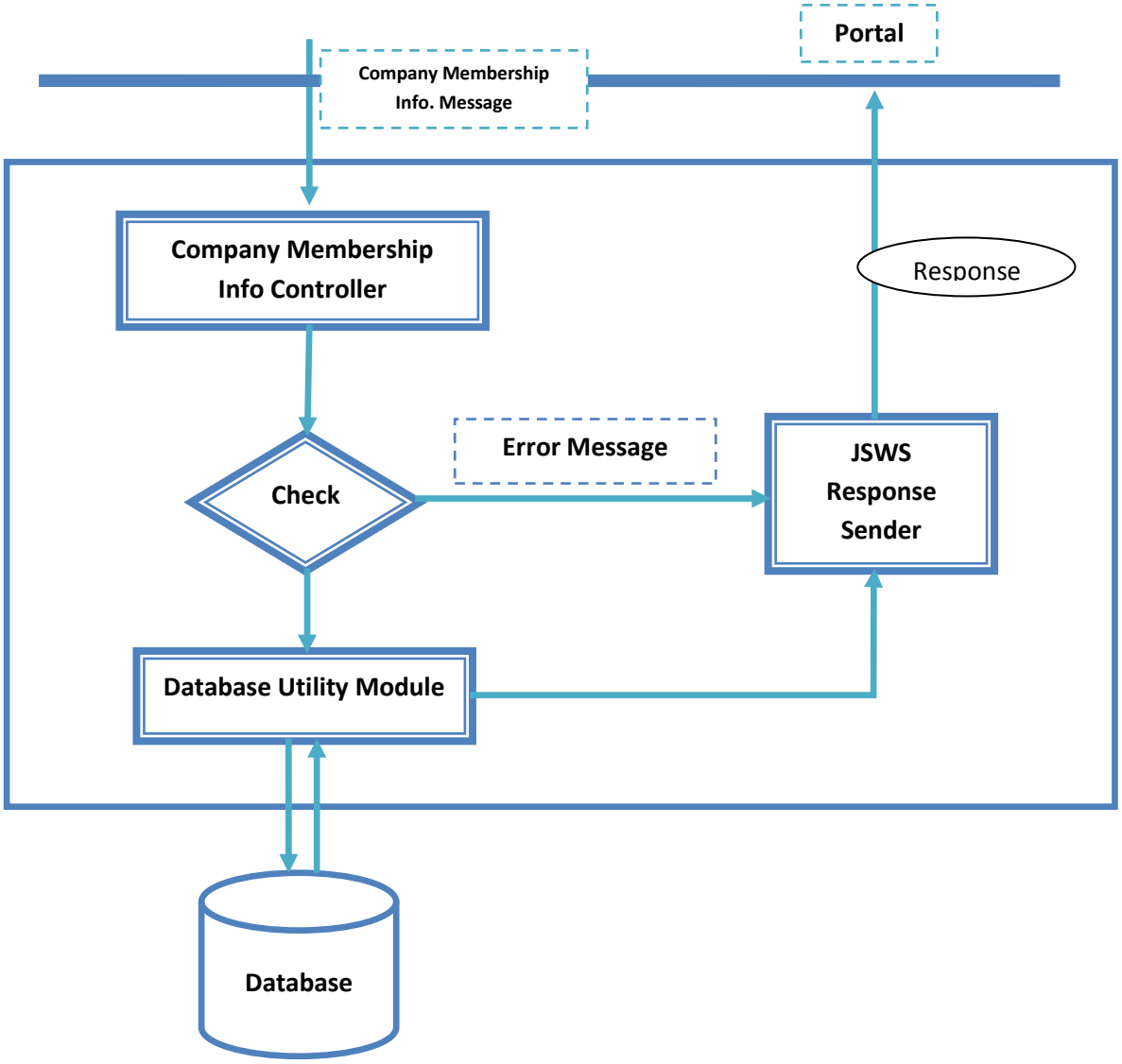


Figure of Company Memership Acceptor Service

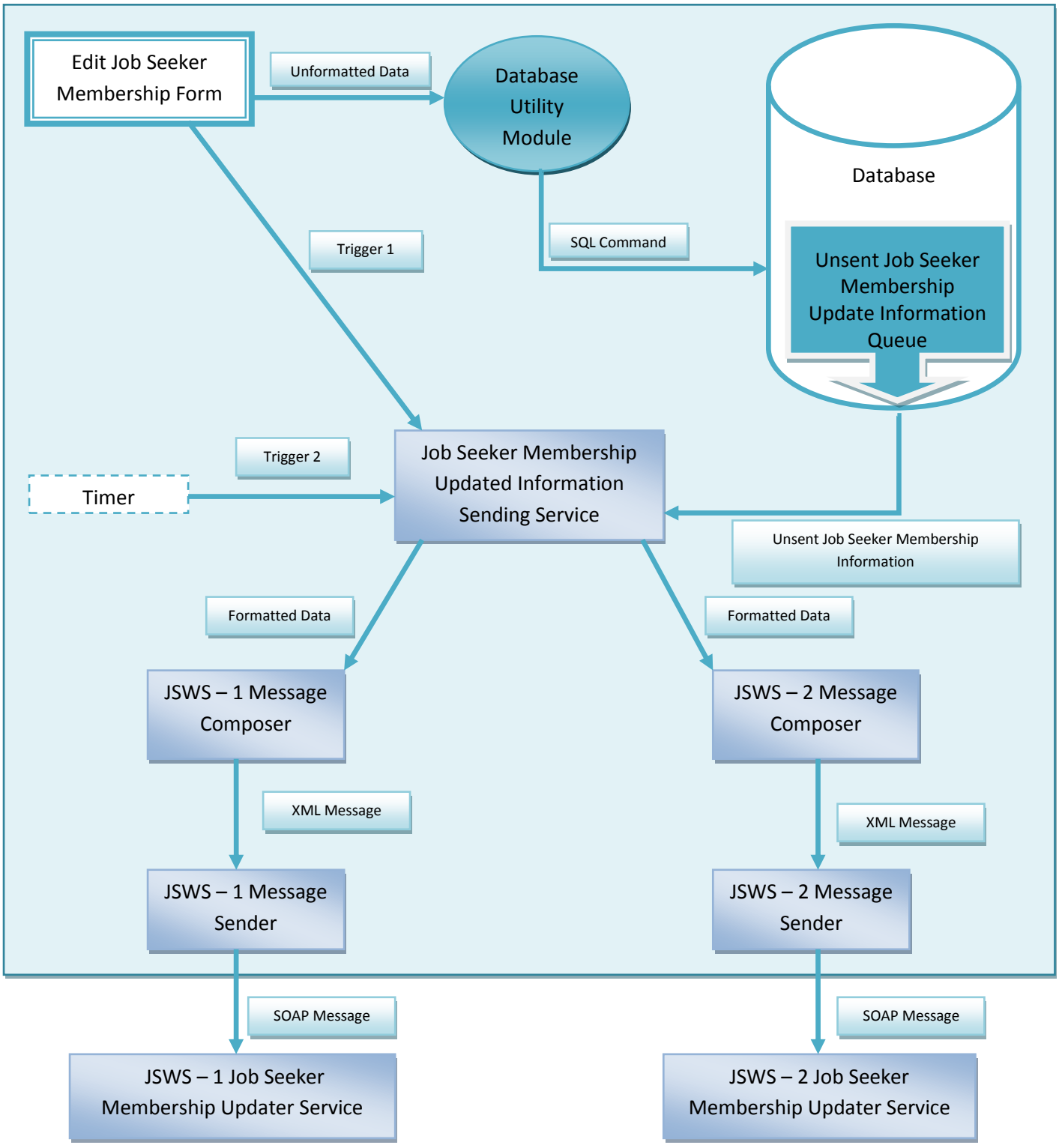


Figure of Job Seeker Updated Memership Information Sending Module

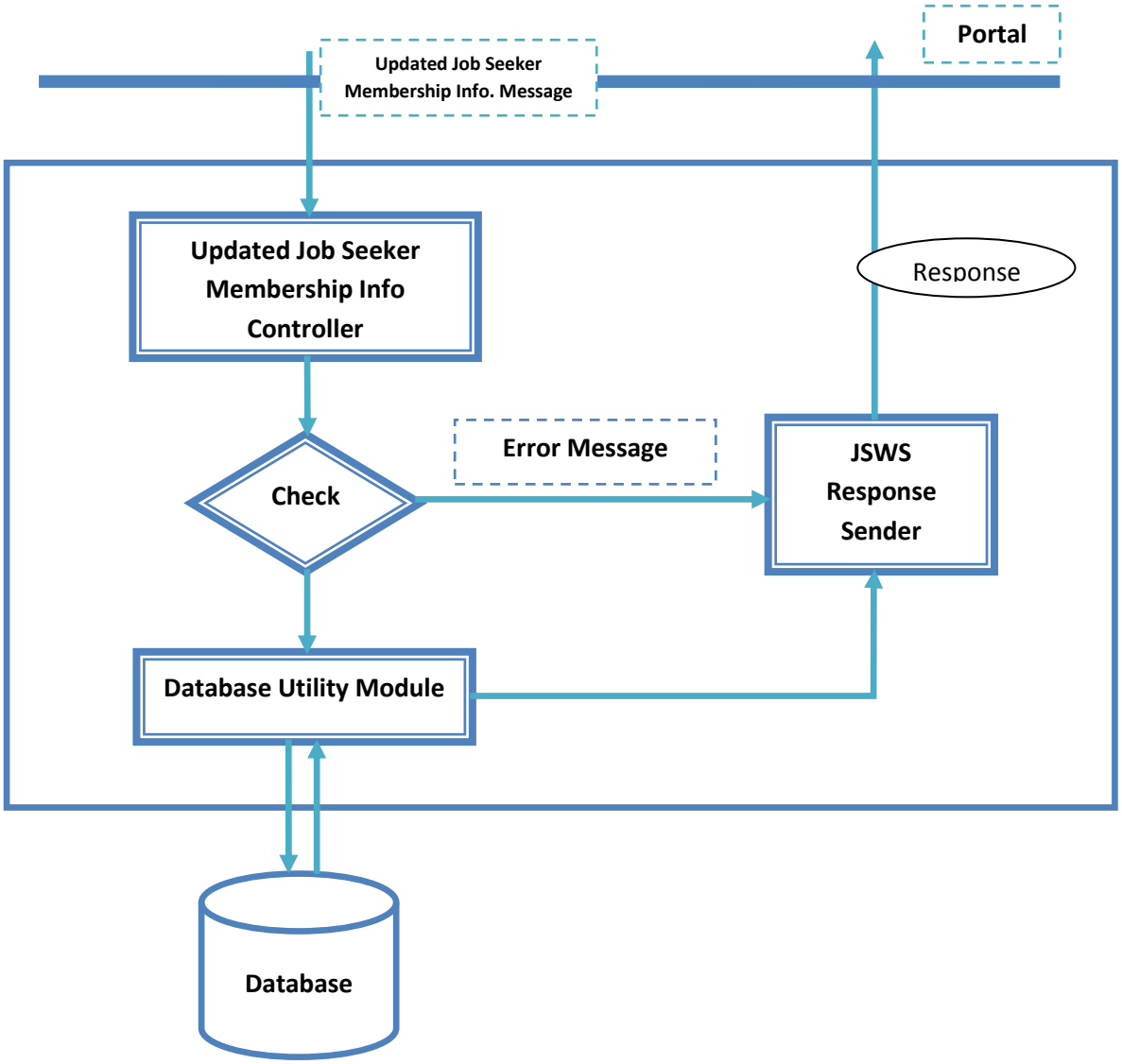


Figure of Job Seeker Updated Memership Acceptance Service

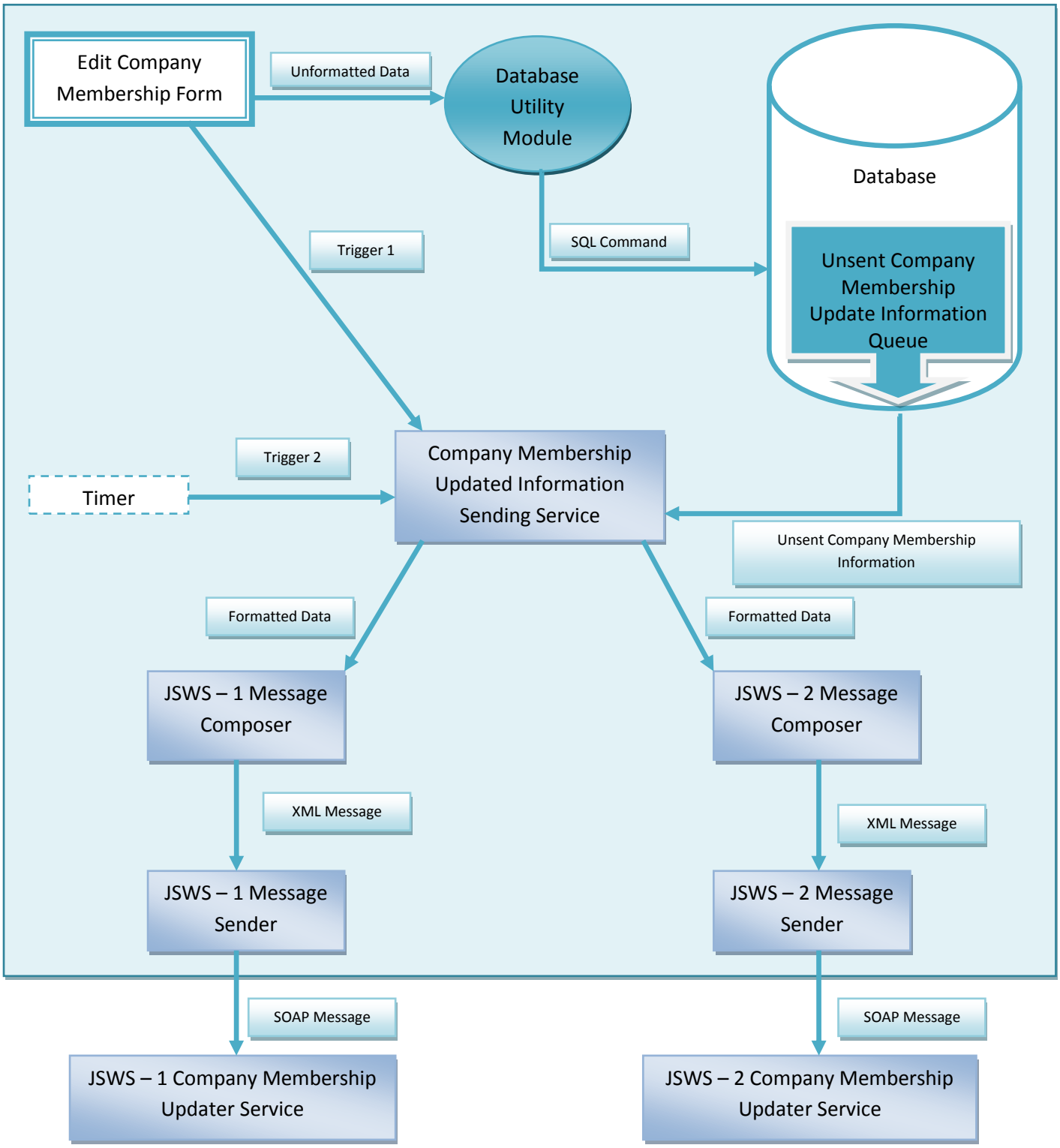


Figure of Company Updated Memership Information Sending Module

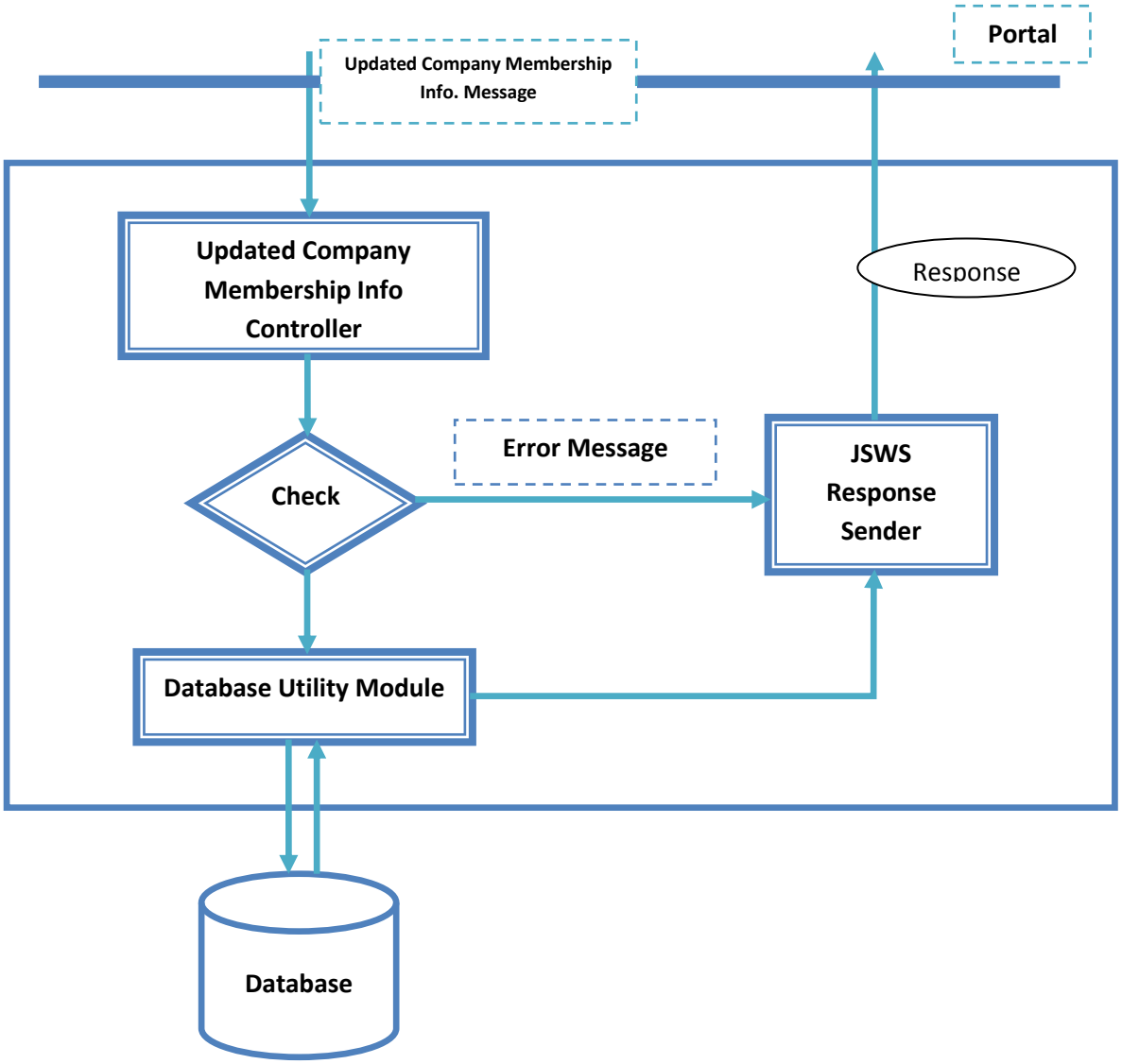


Figure of Company Updated Memership Acceptance Service

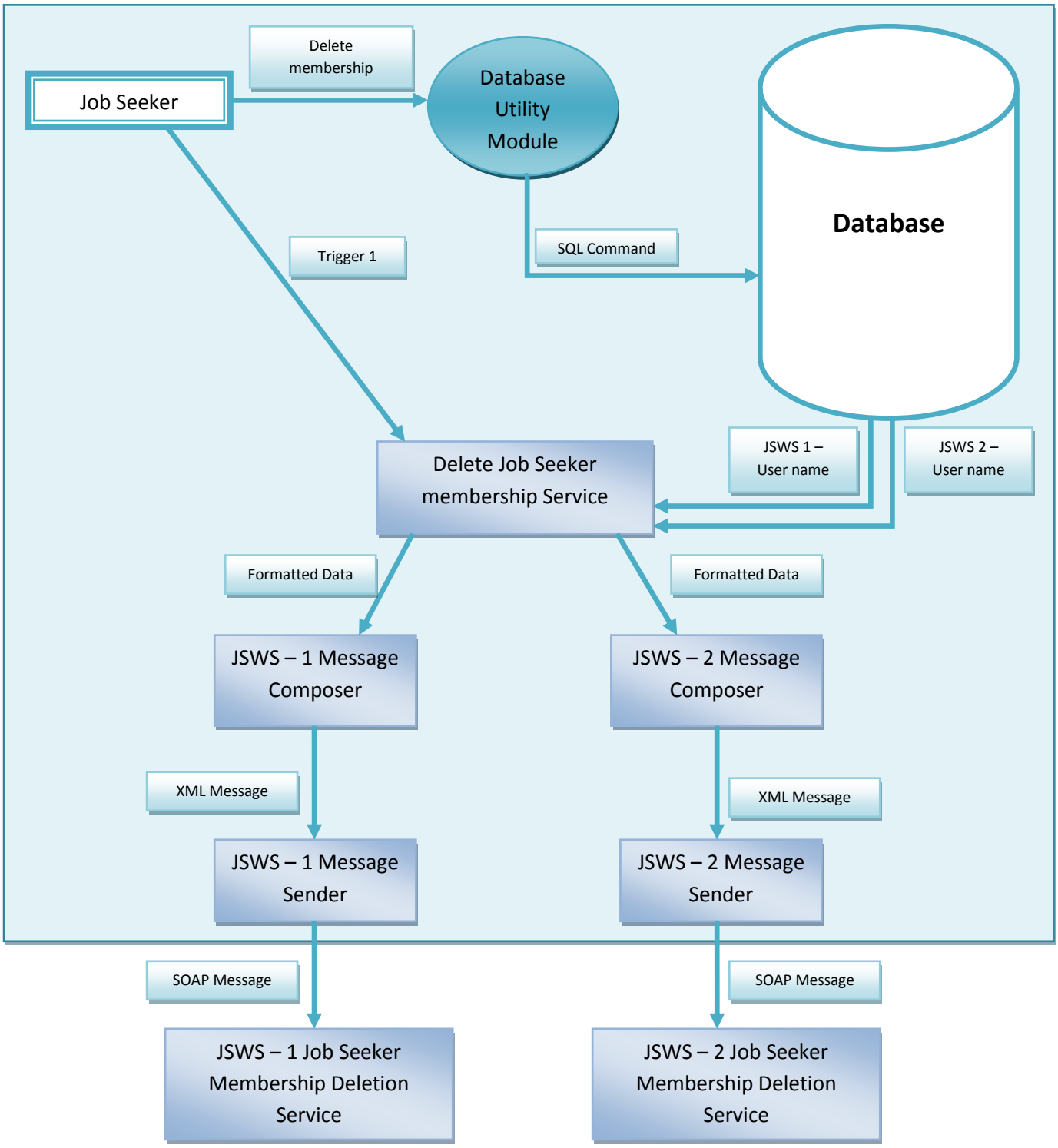


Figure of Job Seeker Memership Delete Request Sending Module

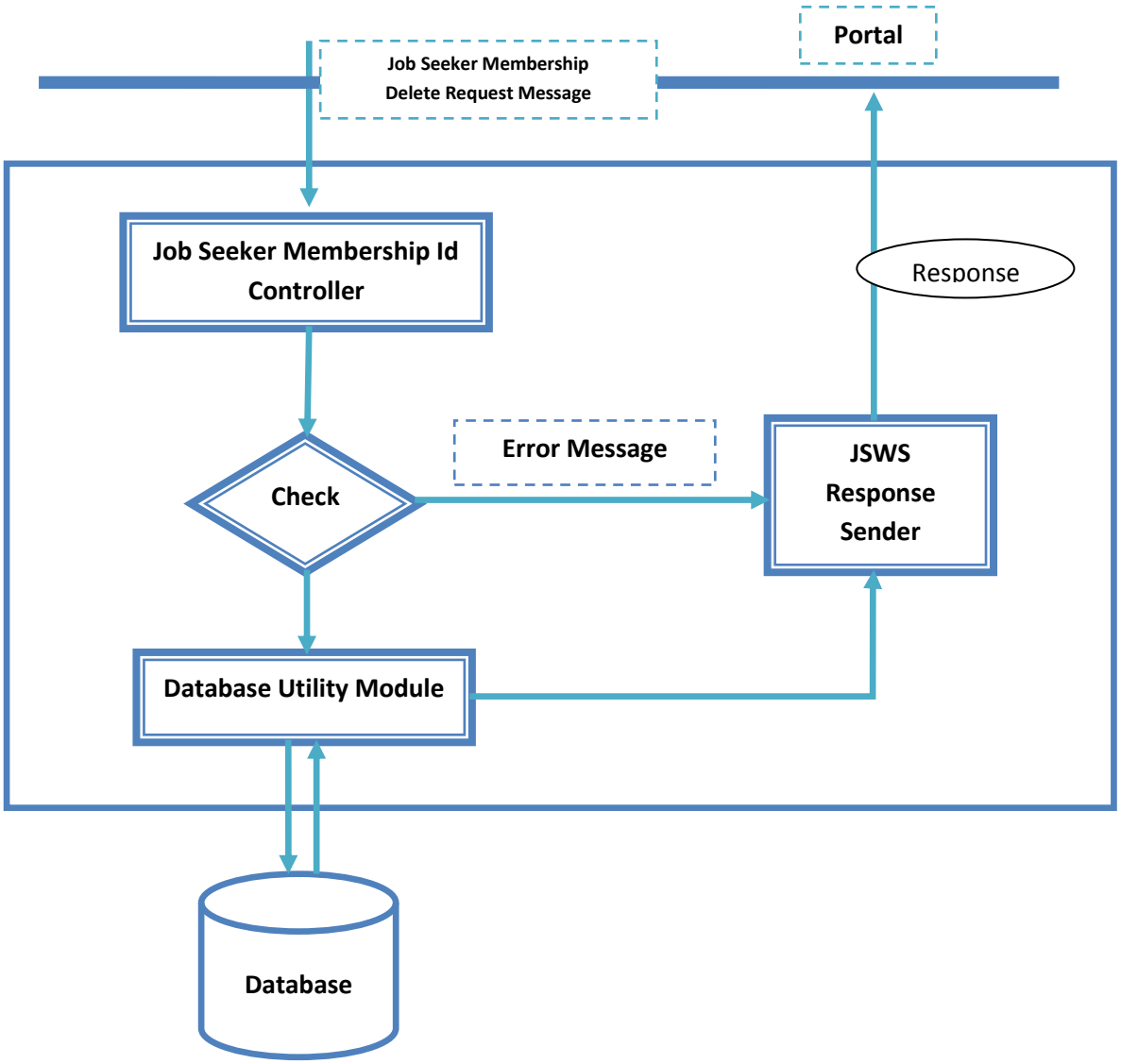


Figure of Job Seeker Memership Deletion Service

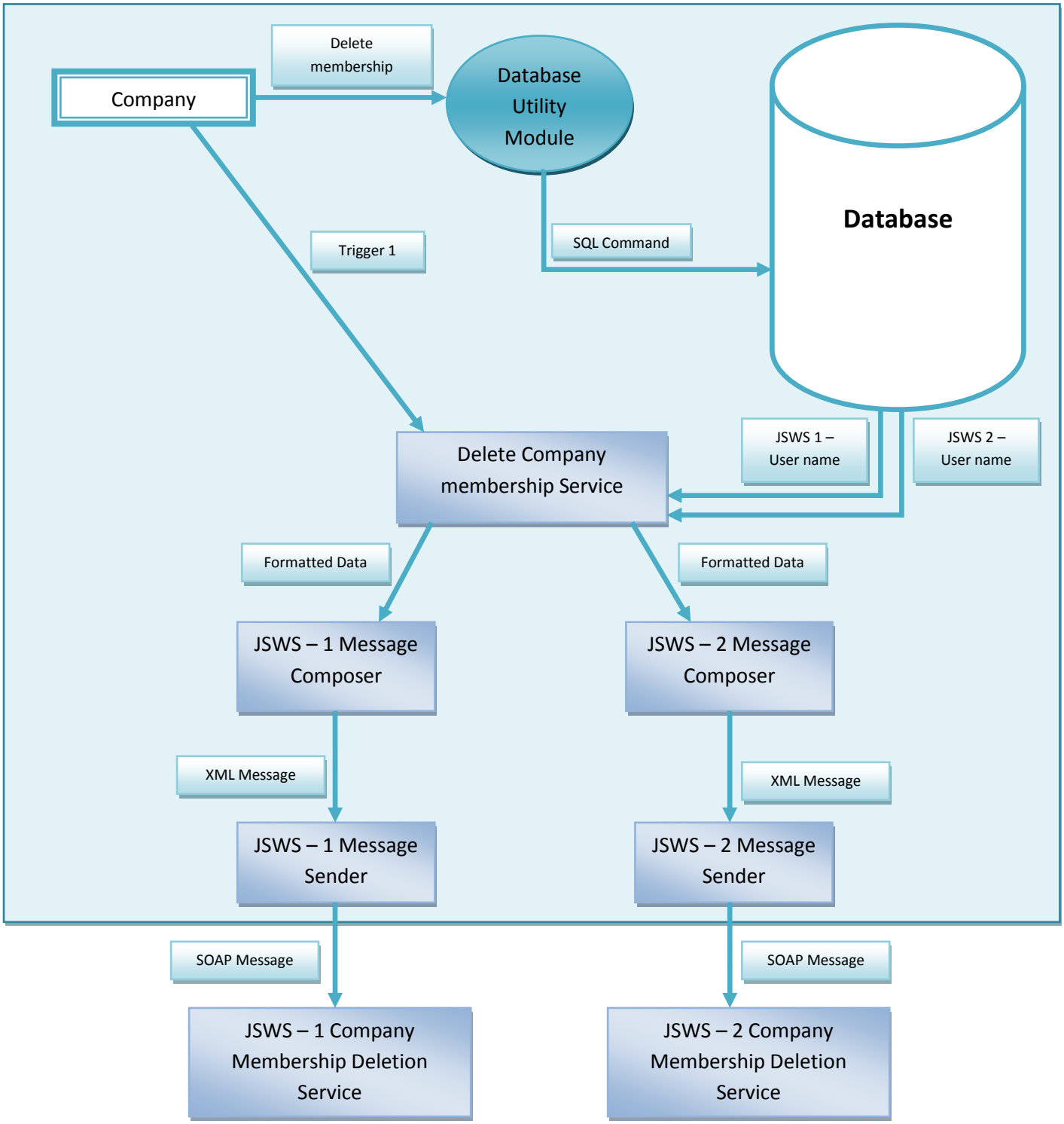


Figure of Company Memership Deletion Request Sending Module

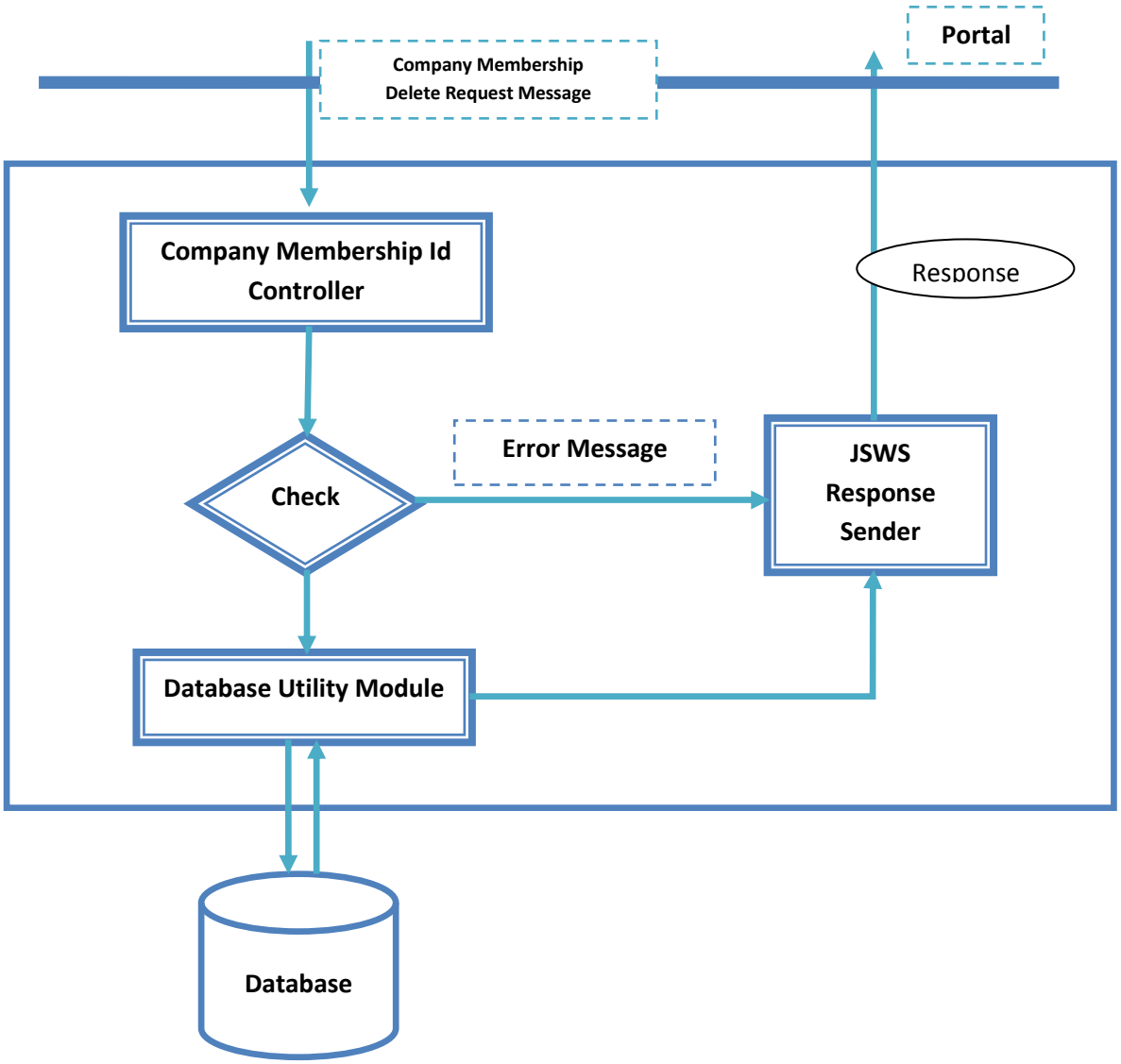
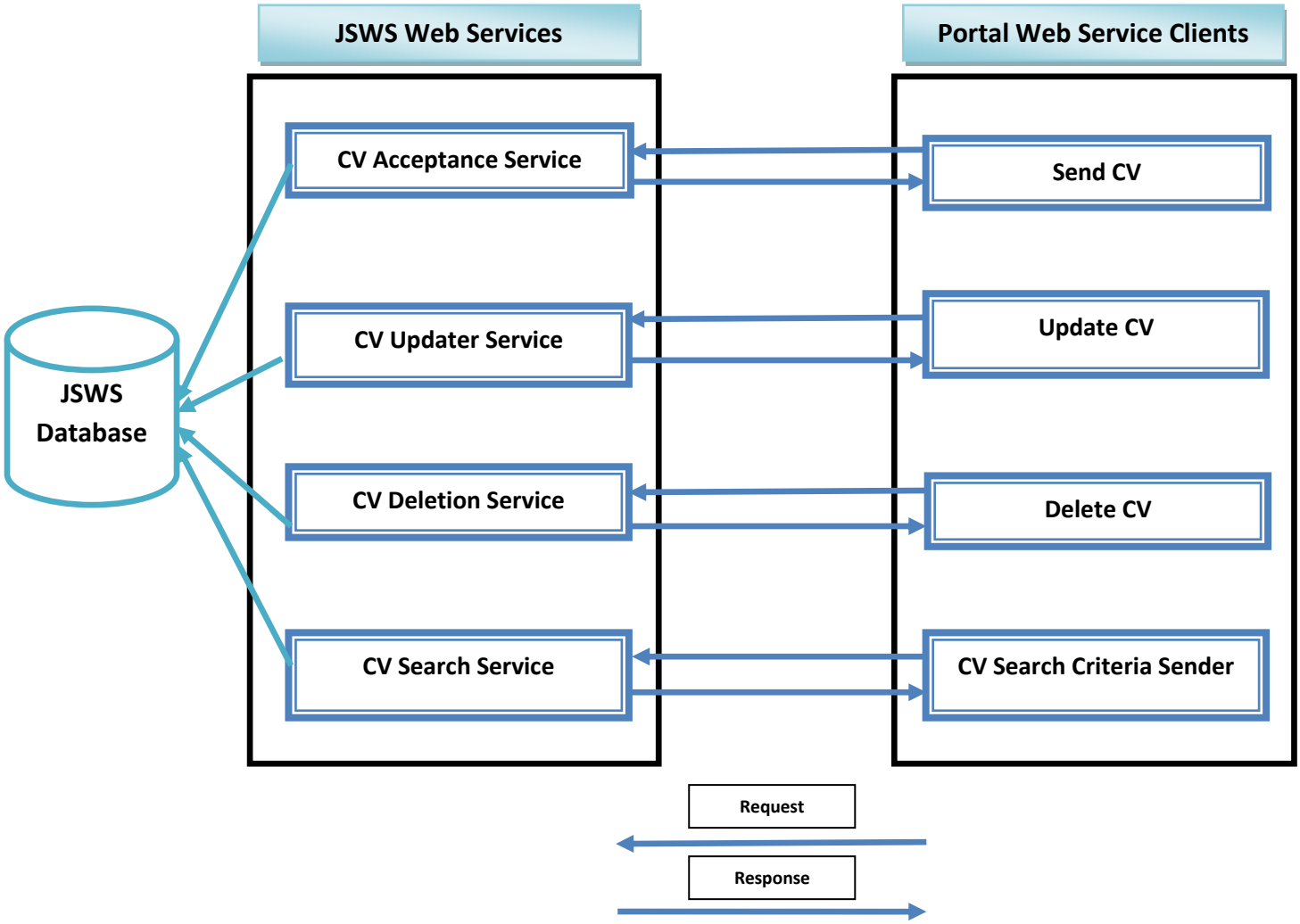
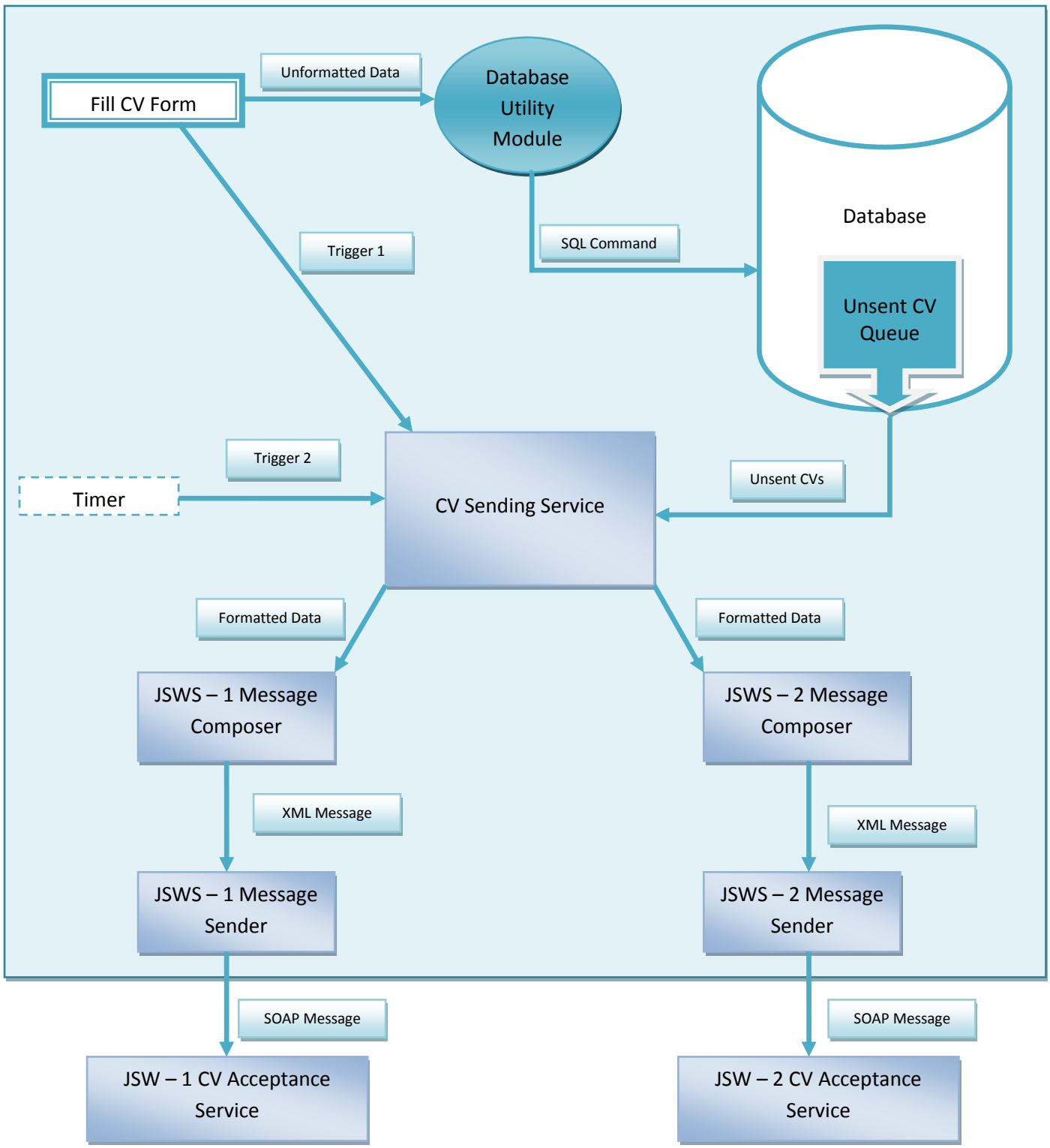


Figure of Company Memership Deletion Service

CV Related Services & Their Clients

The figure below shows the relationship between CV related services and their corresponding clients. The logic is the same as the membership modules.





As you can see from this diagram, when a user fills the CV creation form from our web site, we send this information to our database. In order to add the CV data to our database, we take the "Unformatted Data" by the Database Utility Module. We create insert statements in SQL, to add these data to our database and send the data to our database. When this data is added to our database, we

put the reference of this data to the “unsent CV Queue”. In this queue, with the reference of the newly added data, we also keep in which JSWS this membership has been sent. When there are no left unsent CV’s, we remove the record from the queue. This queue is sent to the job seeking web sites when the CV sending Service is triggered. There are two triggering mechanisms. One is a timer which triggers this service within some interval. The other happens when someone fills a CV creation form. When the service is triggered, it takes the references in the unsent CV queue, takes the related data from the database and starts sending these data to unsent job seeking web sites. During this process, it first takes formatted data and gives it to the message composer. The message composer creates JSWS specific XML message. Later JSWS specific message sender takes this XML and converts it into a SOAP message. This SOAP message is received by the web service which accepts this CV.

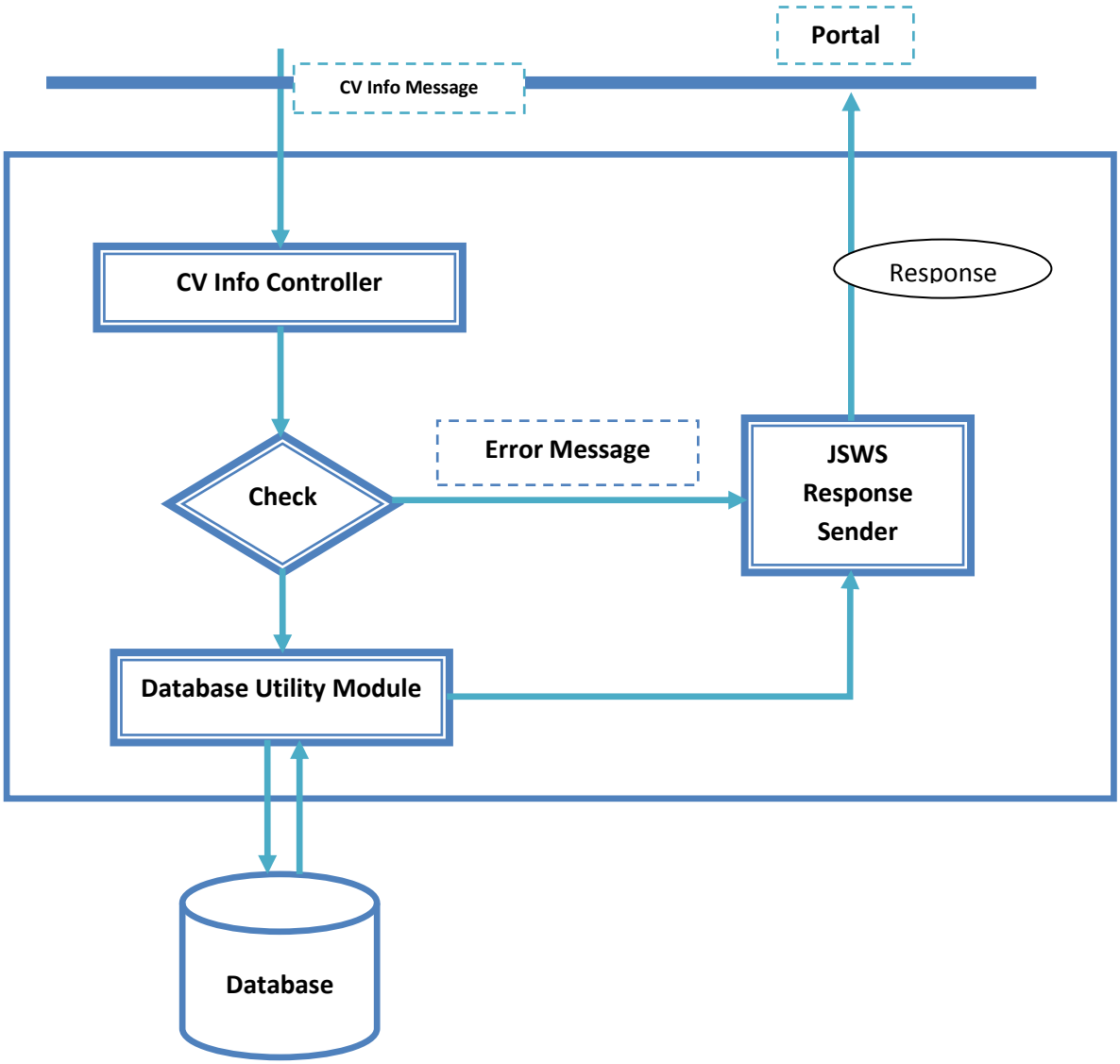


Figure of CV Acceptance Service

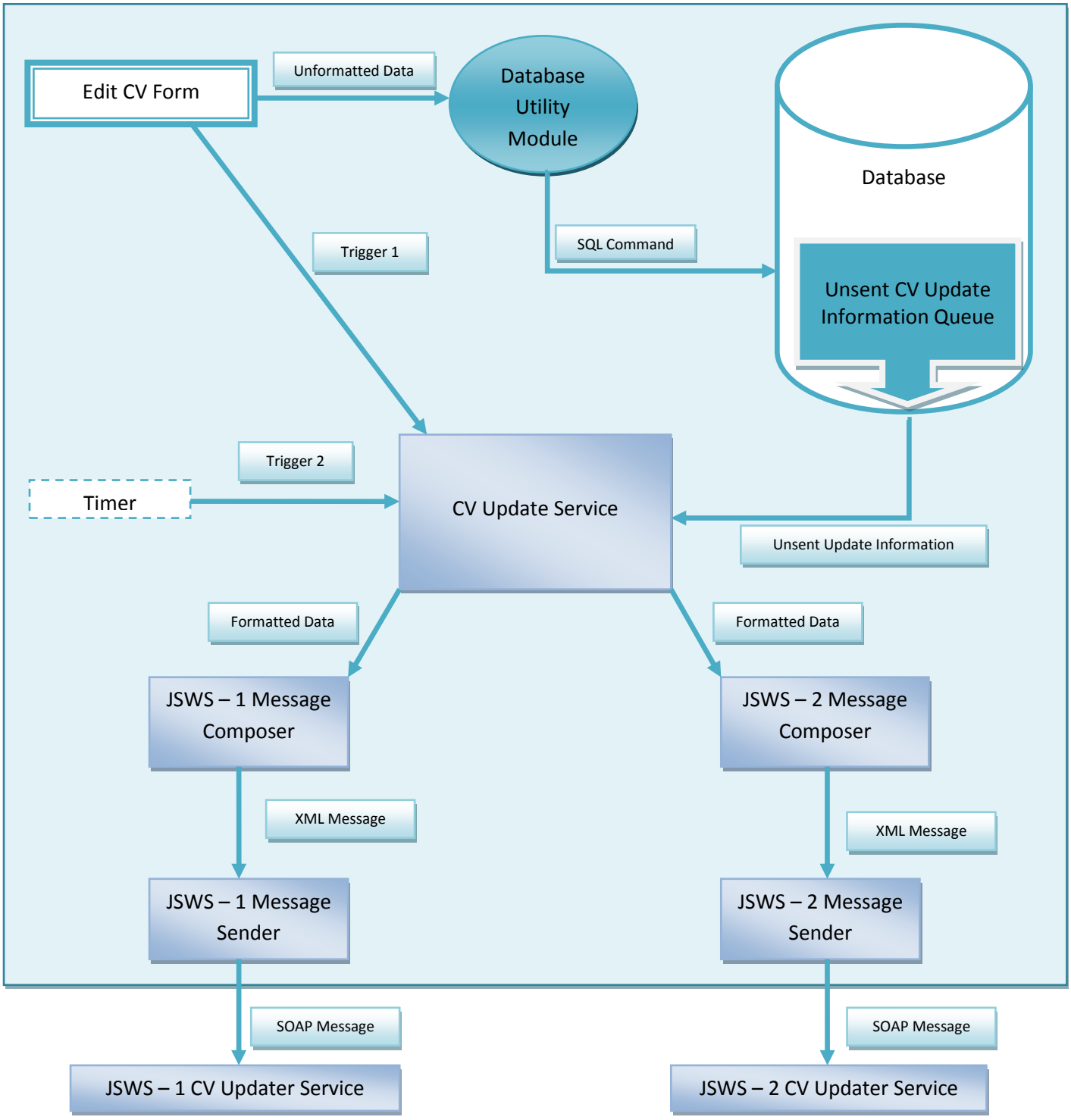


Figure of Updated CV Information Sending Module

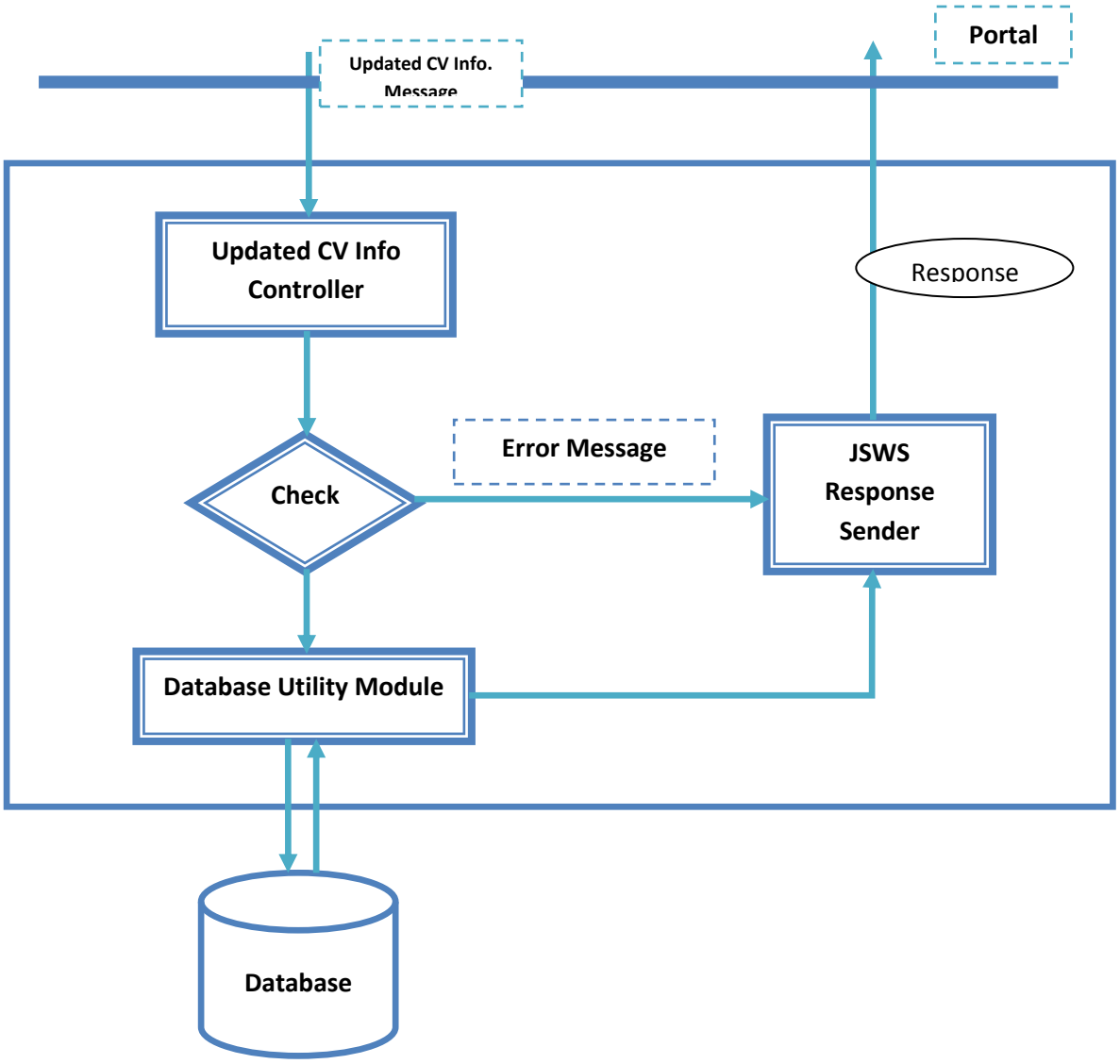


Figure of CV Updater Service

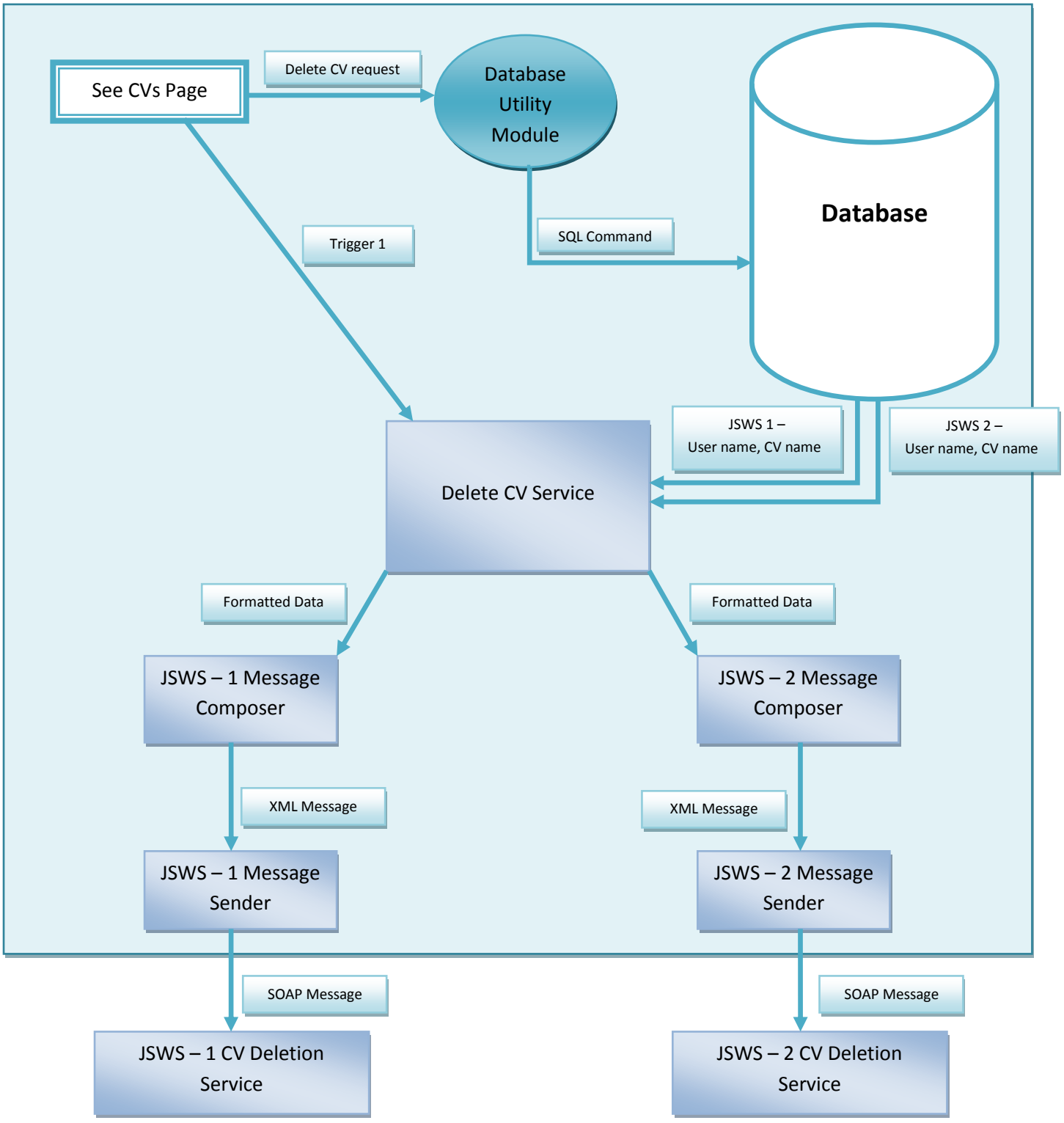


Figure of Delete CV Request Sending Module

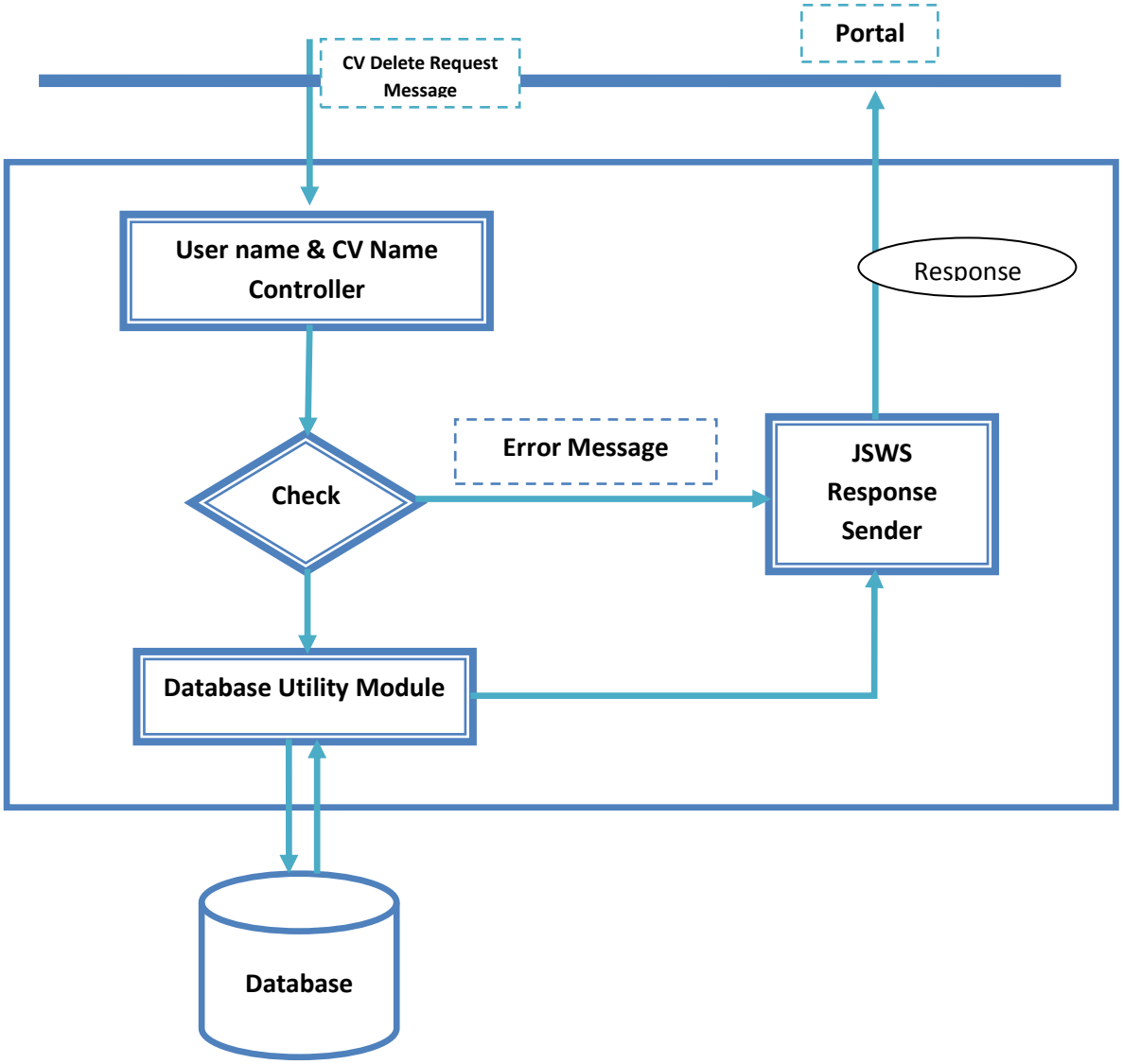
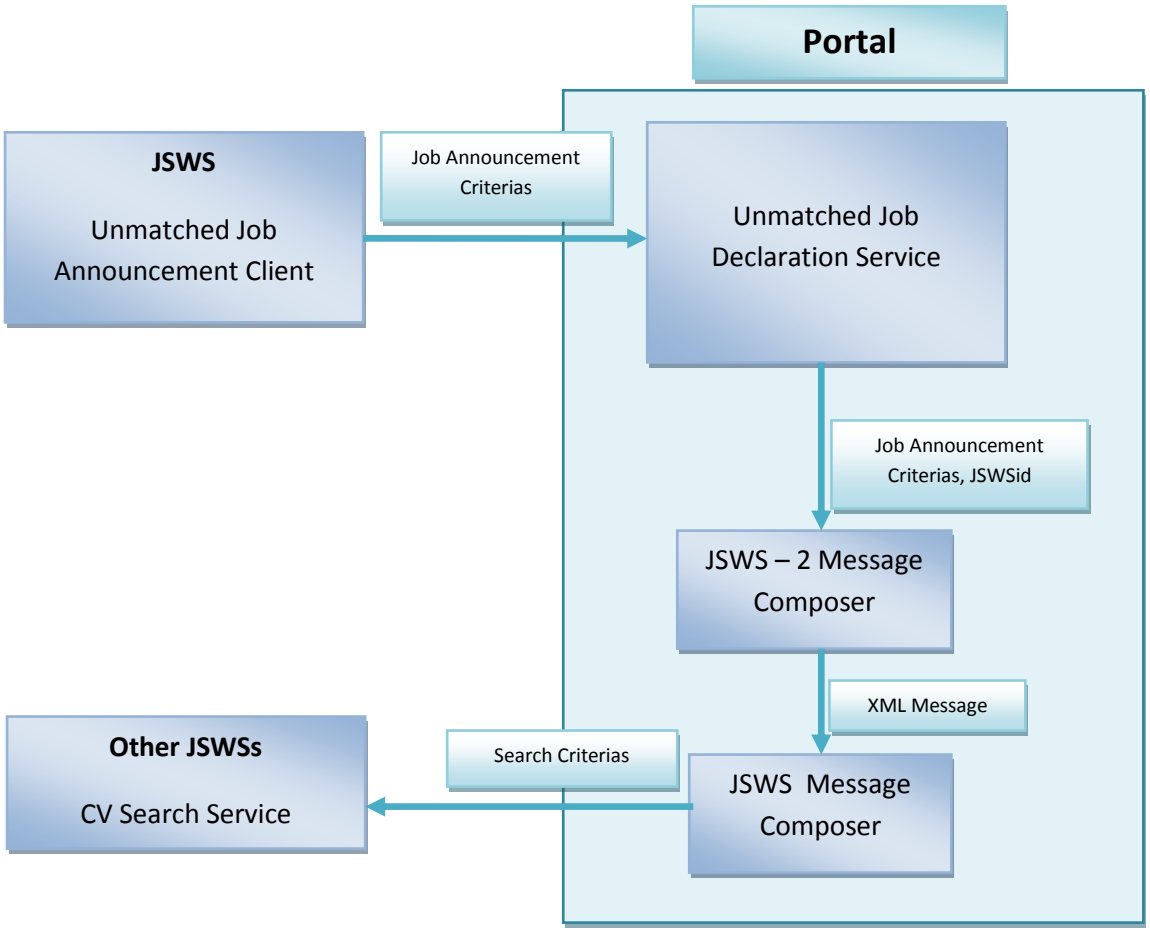


Figure of CV Deletion Service



This figure needs some explanation because of its difference from other diagrams. In fact, in this case our portals Unmatched Job Declaration Service is in a position of "Service provider", and the JSWS that sends Job Announcement that has no matched CV, is in a position of Service consumer as it requests us to search for appropriate CV from other JSWSs working under agreement with us.

Any career Web site working under agreement with us can send job announcements that no CV applied for, manually or periodically. In this case, our Web service takes job announcement criterias and composes an XML message for other JSWSs to search CV for. JSWS Message Composer produces one type of message, because this situation is not so related to their database structure. JSWSs will take the message as array of criterias and search for matching CVs as the figure below shows:

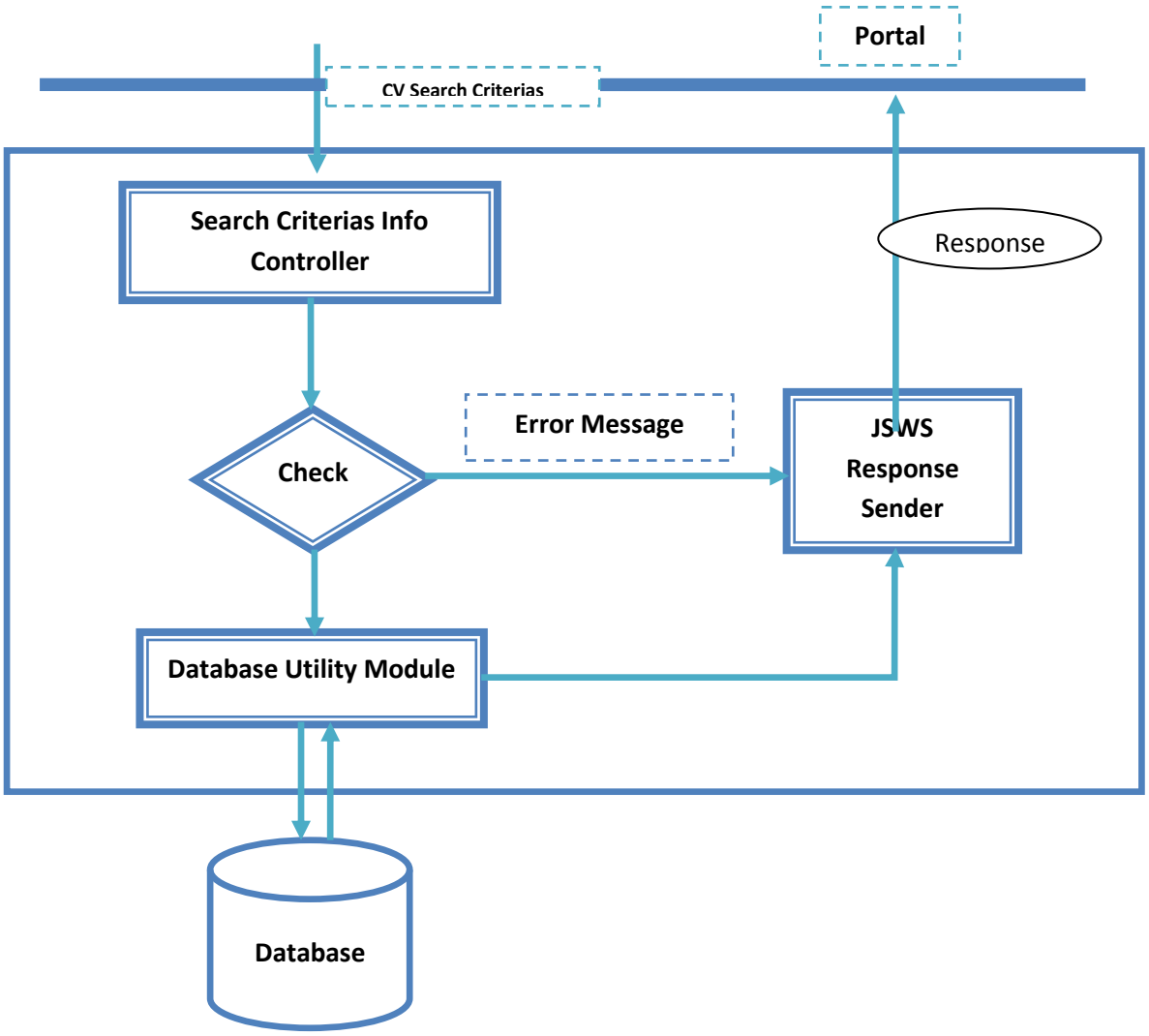
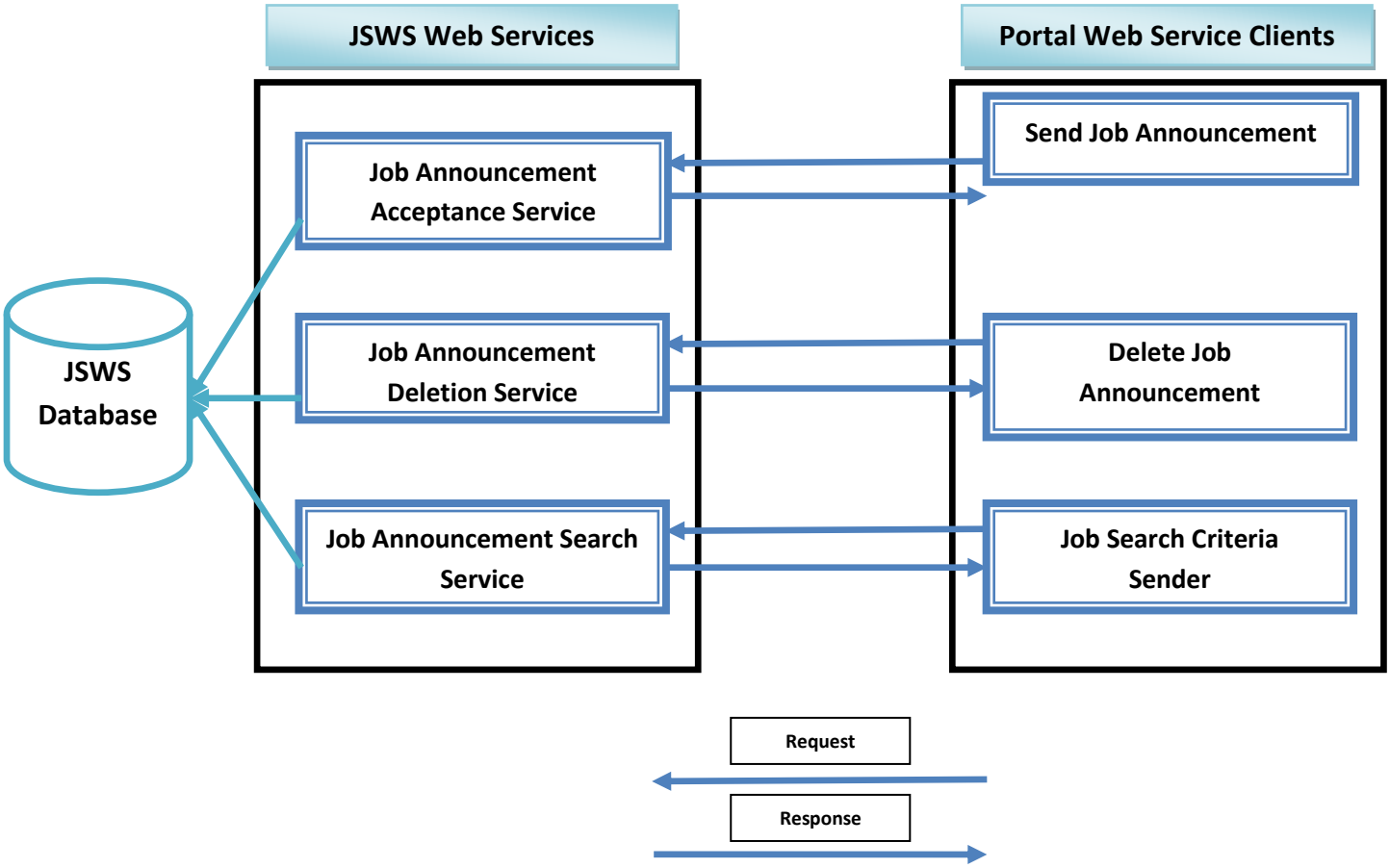


Figure of Search CV Service

Job Announcement Related Services & Their Clients

The figure below shows the remaining inter relationships between Web services and their clients.



The logic behind the communication is still the same. Clients send request messages and the service providers respond that request. However, there is no update service in the JSWSs in our design, so no update option is provided for the companies for a Job announcement after it is sent to JSWSs. Because if there were, companies will be able to create new job announcements continuously just by updating the ones previously sent.

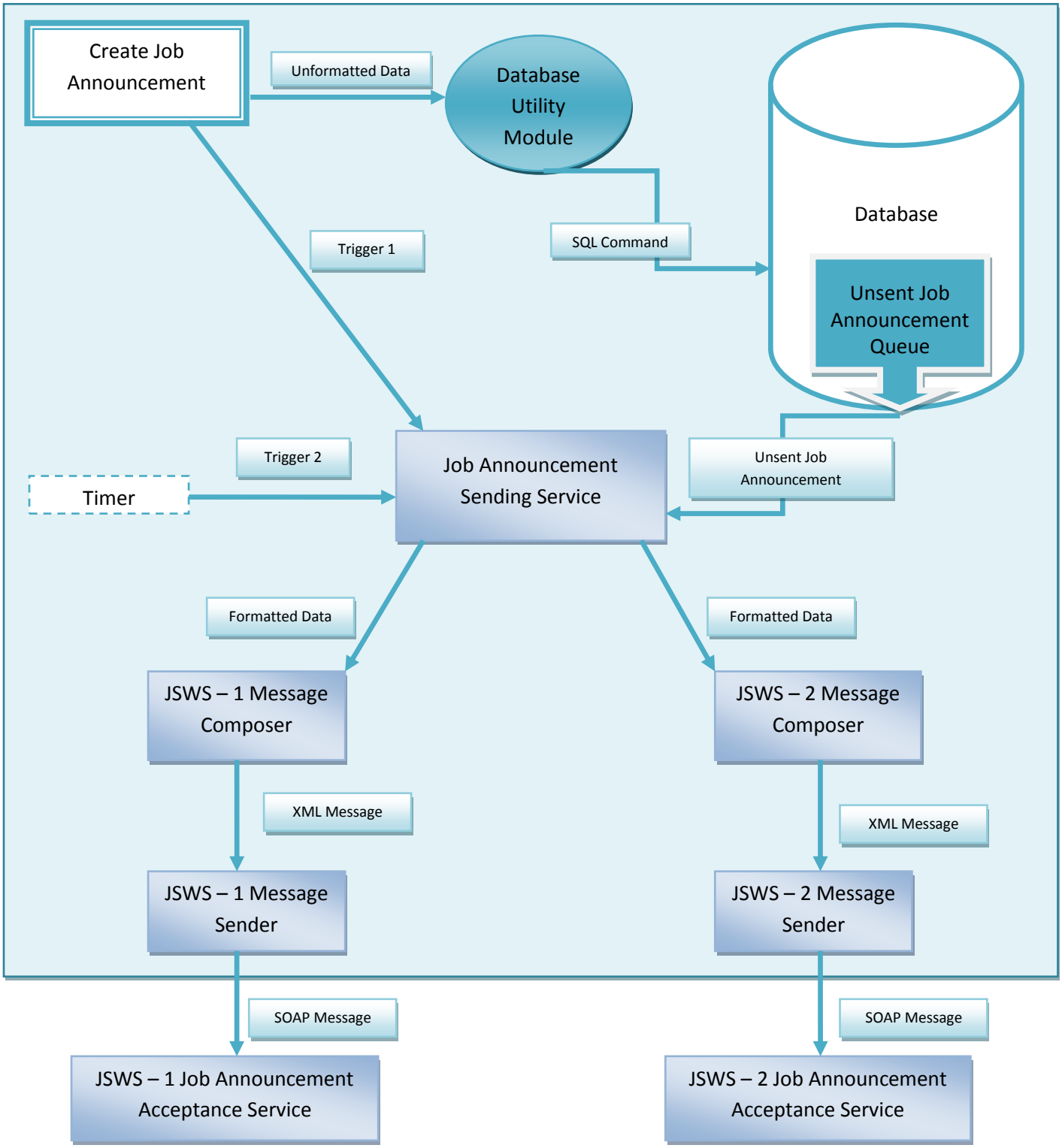


Figure of Job Announcement Sending Module

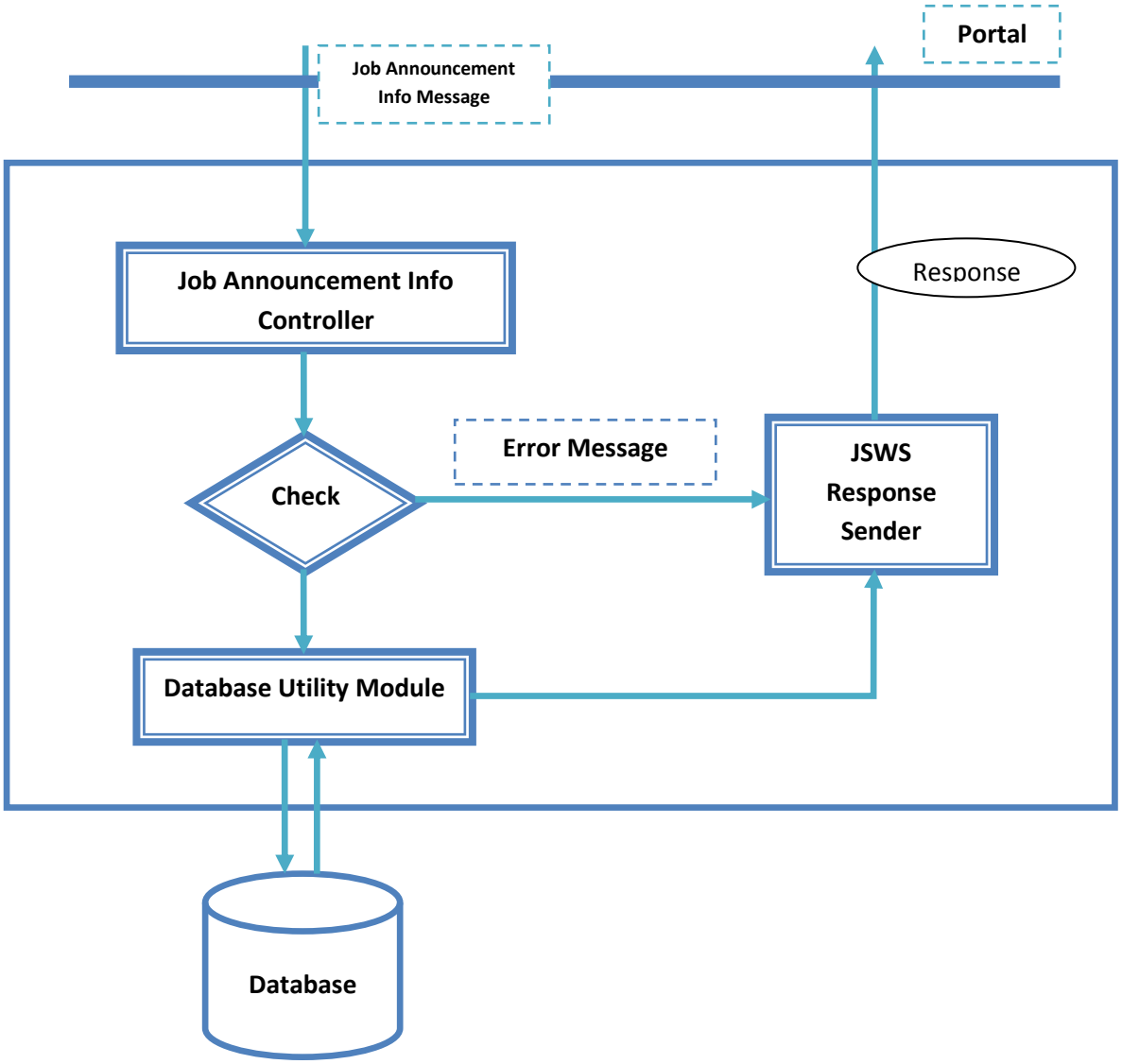


Figure of Job Announcement Acceptance Service

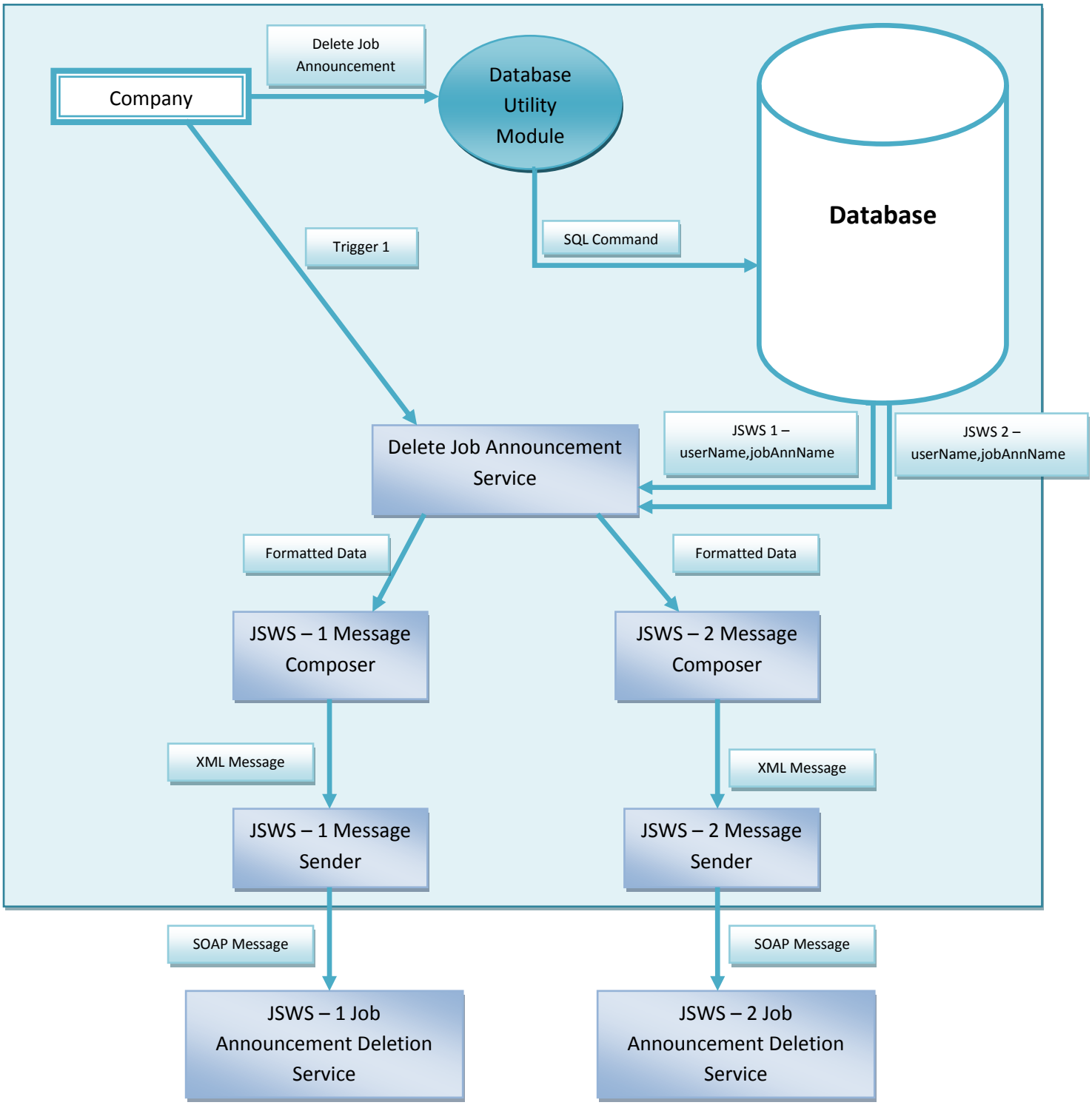


Figure of Job Announcement Delete Request Sending Module

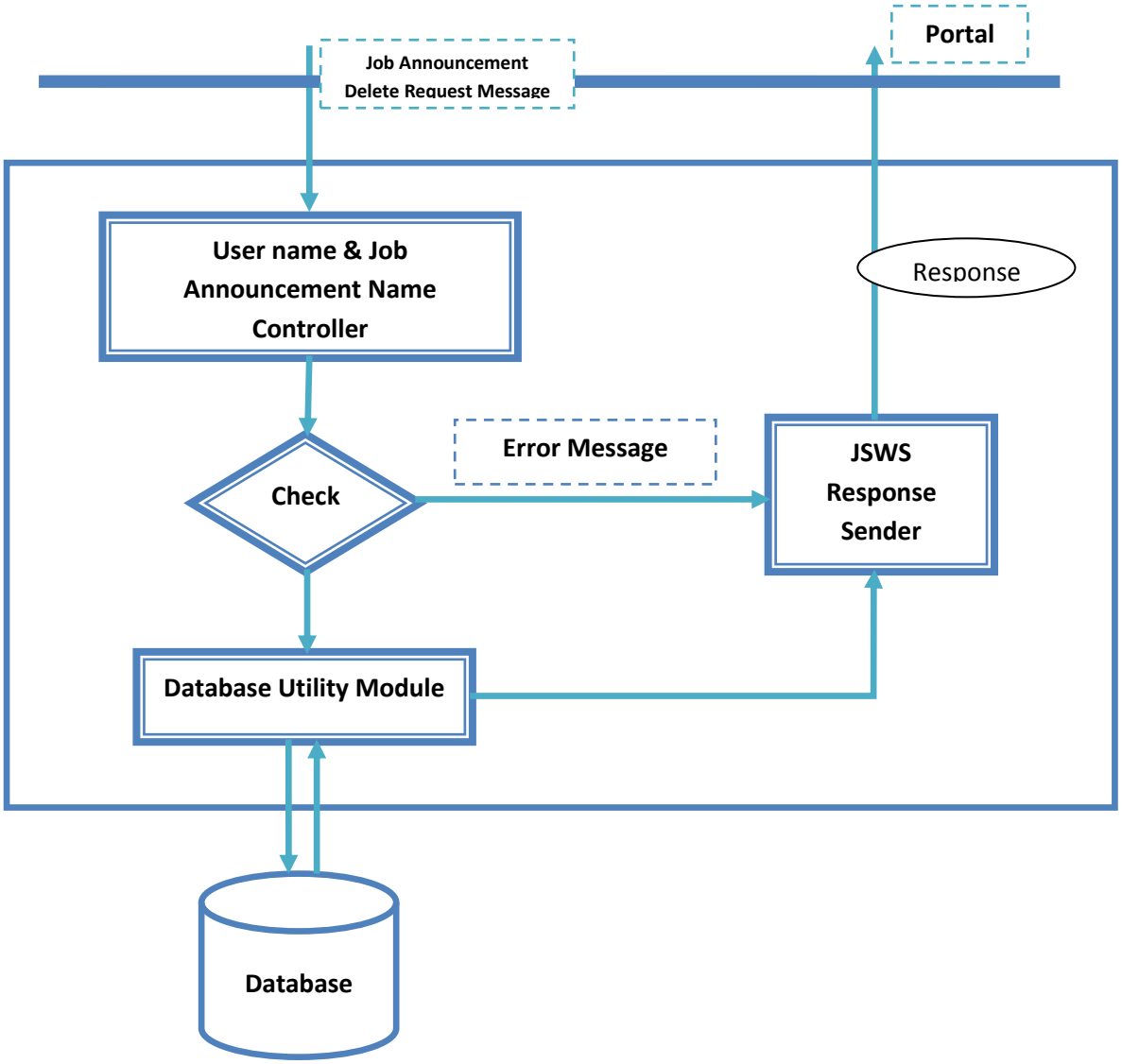


Figure of Job Announcement Deletion Service

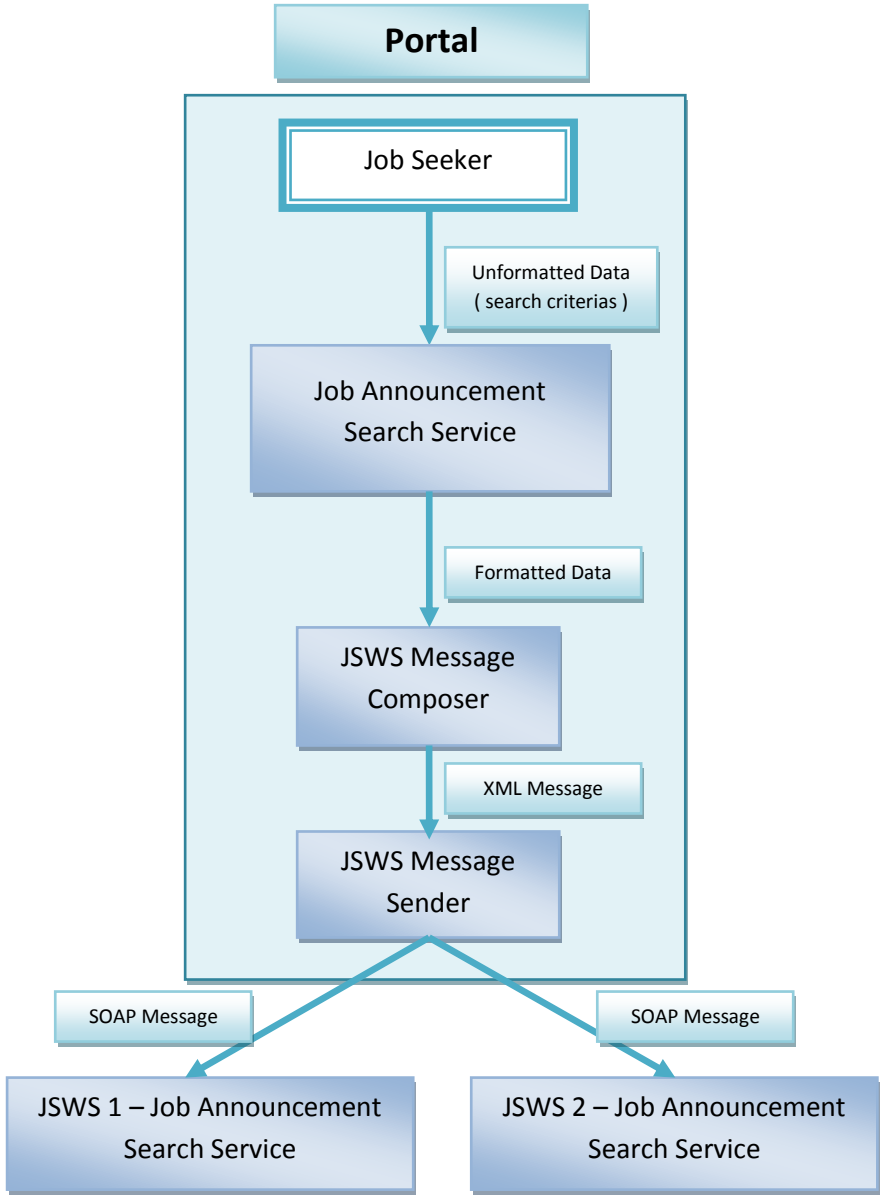


Figure of Job Announcement Search Criteria Sending Module

As we stated in the system properties, anyone will be able to search for available jobs from our portal without registration. When a person wants to see job announcements having some specific criterias, he will give the necessary specifications and our Job announcement service will take those

specifications from the user interface components composes a well formed message. After that, JSWS message composer takes that well formed message and converts it into XML message and passes it to the JSWS Message sender module which is the client of the “Job Announcement Search Web Service”.

As in the CV search criteria sender module, there is only one JSWS message composer because of the same reason.

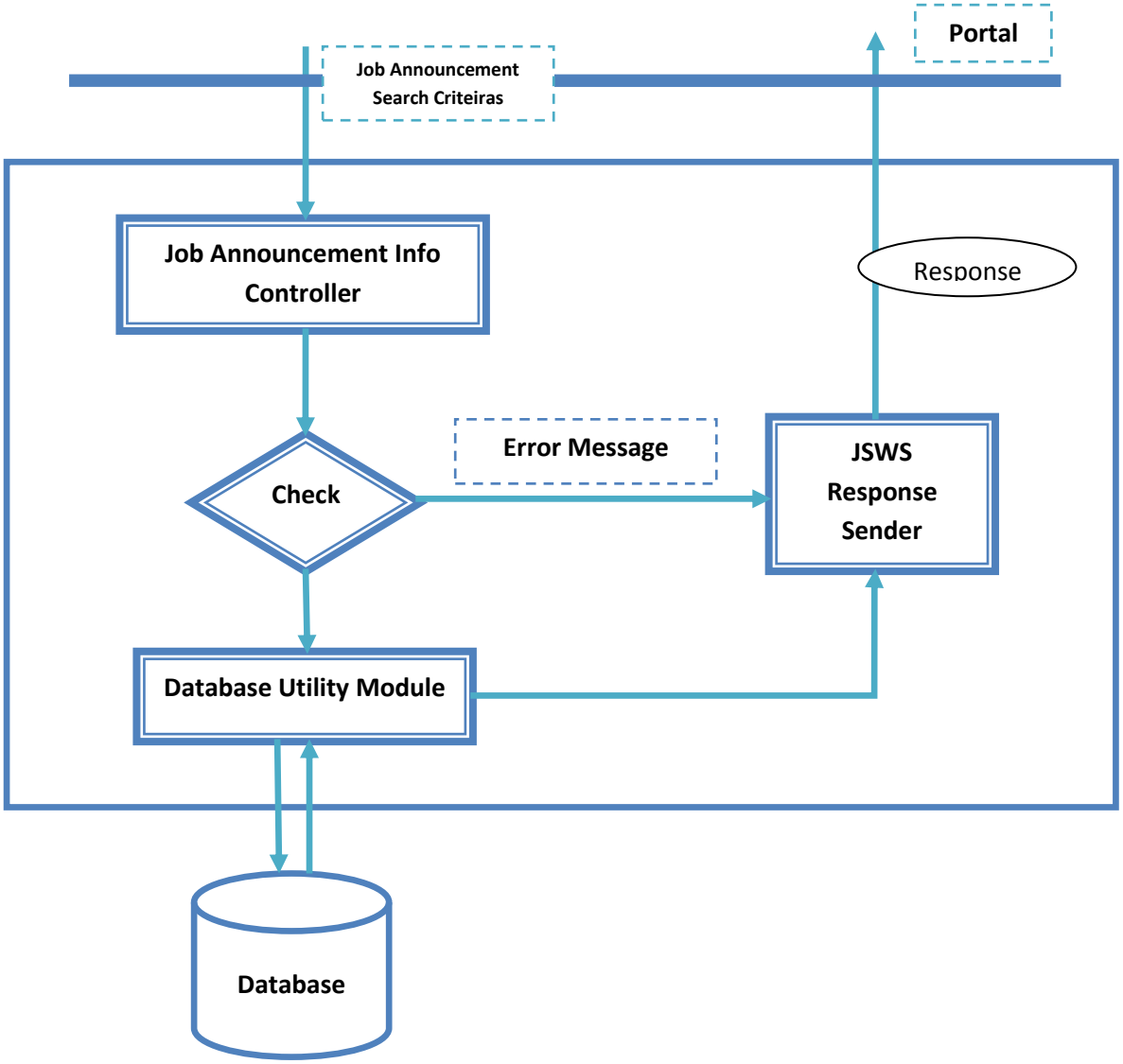


Figure of Job Announcement Search Service

3.4. DATA EXCHANGE BETWEEN SERVICES (MESSAGING)

Sending and receiving data is done with exchanging XML formatted messages. There are several “Message Exchange Patterns” (MEP) for data exchange between Web services and their clients. We will use “Request and Response” pattern which the most widely used pattern. However, there are two very important issues about messaging:

- **Reliability, and**
- **Security**

After a service transmits a message it should know whether the message successfully arrived at its intended destination, or failed so that it requires a retransmission.

A core part of reliable messaging is a notification system, in other words sending acknowledgement to the message source. We will design our Web services in a way that whenever they receives a message, they will send back a notification to the message source indicating they received the sent message. And a time out mechanism will work, so that the service can decide whether a retransmission of the message is necessary or not.

Security requirements for automation solutions are nothing new to the IT world, and since this project is based on transferring messages between services, security becomes one of the most important considerations. For a service requestor to access a secured service provider, it must first provide information that expresses its identification.Once the identification information is authenticated, the message recipient will need to determine what the requestor is allowed to do, which is “authorization”.

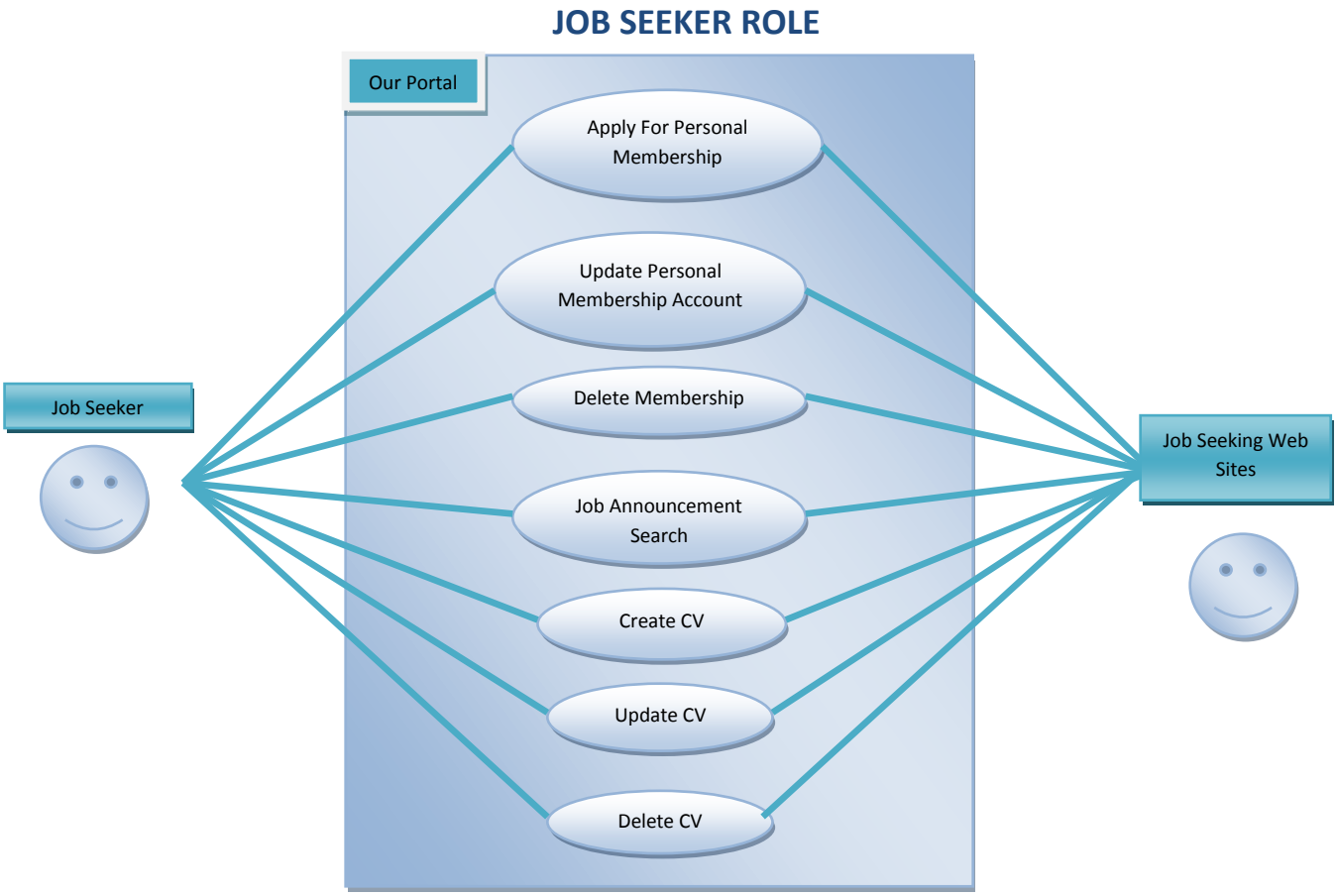
There are some standards about reliability and security which are WS-ReliableMessaging and WS-Security, XML-Signature, XML-Encryption. Apache plug-ins will help us to catch those standards. “Apache Sandesha2” plug-in will provide WS-ReliableMessaging and “Apache Rampart” will provide WS-Security. As we go in deeper within this plug-ins, we will give more detailed information about them in the implementation step.

3.5. A COMPOSITION OF SCENARIOS IN THE SYSTEM

At this part of the report, we will explain the scenarios of our system. By scenario we mean who can do what by using our portal. For this aim, we have changed the Use-Case diagrams we included in our previous reports. The diagrams we include below don't fit to the definition of use case diagrams but they are similar in aim. In all of these diagrams, we use the phrase "who (1) can do what (2) affecting who (3)". To make this definition more clear, let us define the words we have numbered:

1) By this word, we mean the starter of the scenario. These are the users to the left of the system box.
2) By this word, we mean the action done by using our system. These operations are defined inside the user module.
3) By this word, we mean the affected user of the system as a result of the connected action. These are the users to the right of the system box.

Now we will give the detailed definitions of the operations that can be done by using our system.



A job seeker can apply for membership, update and delete the information in his account, make some job announcement search and create, delete or update a CV belonging to him. The effects of these operations are all seen in job seeking web sites that we are in collaboration with. Now we will describe these operations in detail.

1) Apply For Personal Membership:

A personal user can create an account from our web site by using filling the form under our web sites personal users tab. When the user declares that he wants a user account by clicking “Üyelik Oluşturmak İçin” button, we direct the user to the new member form page. After the user fills the form and clicks submit, we create the account in our web site and then try to create the same account from job seeking web sites. We try to find unused names in job seeking web sites. For example, we first try “portal_username_1”. If this name is available in job seeking web site, we use this name. Else we increment the number at the back and try again until we find another. We guarantee that the account will be created from all job seeking web sites when a user requests this from our portal.

2) Update Personal Membership Account:

A personal user can update an account that he created from our portal by using this property. The property can be reached when a user has logged in using his user name and password. When the user goes to the update page, forms filled with users previous information will be provided to the user. The user will then change the fields he wants and submits the changed form. The required updates will be made similar to adding a personal account.

3) Delete Membership :

A user may withdraw his account when he no longer needs it or when he is not satisfied with our service. For this aim, he just logs in to our portal with his user name and password and then declares that he wants to cancel the account. After a confirmation step as this declaration we cancel the accounts that we have created from our portal and job seeking web sites.

4) Job Announcement Search :

This service of our web site can be both by the users who logged in or without logging in. For this aim, we provide some forms defining the criteria of the job to be queried. After filling and choosing these preferences, the user submits these preferences. We query these criteria from all the job seeking web sites we collaborate with and return the results to the user as a list of information and links.

5) Create CV :

A user who is logged in to our portal can create new CVs by filling forms. A user may have more than one CV with different information. For this reason, we don't refer to users previous CVs and show them a prefilled form. We give users a completely empty CV creation form which is standard for all the users. When a user submits his CV, we add this CV to our database and also distribute this CV to the job seeking web sites. This information is kept under the users account information as you can see the details of this relation under the database definition part of our report.

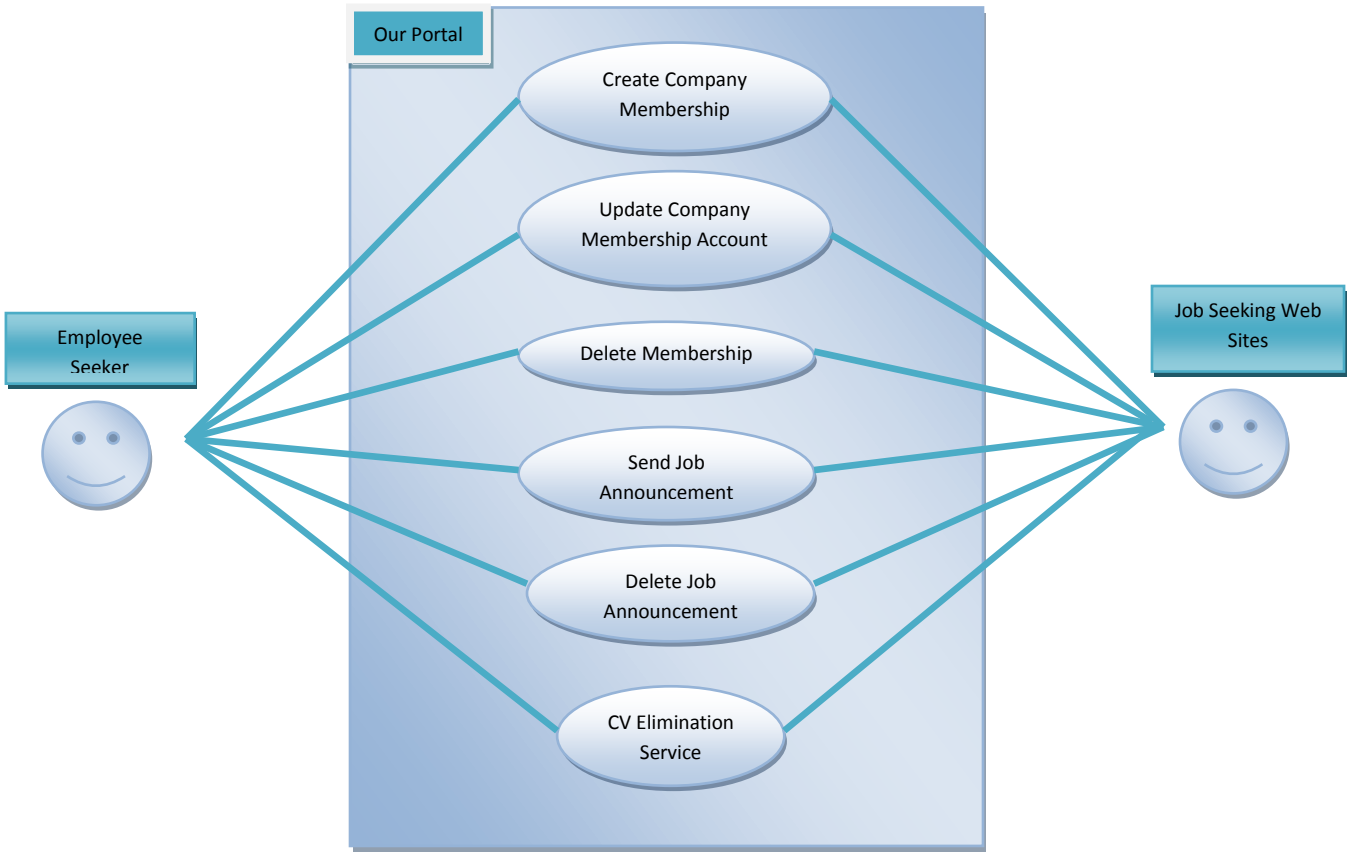
6) Update CV :

A CV may be out of date after a while. Instead of deleting an out of date CV and creating a new one, a CV may be changed partially. For doing this, a user declares that he wants to update a CV. Our portal provide him the CVs that user owns. User selects one of these CVs and a form filled with the information inside the CV is brought to the screen. User can easily change the information he wants by this screen and commit the changes. The changed information is changed from both our portals database and the job seeking web sites database.

7) Delete CV:

A user may want to delete one of the CVs that he created. So for this aim, when a user want to delete a CV, we provide them the list of CVs that user has created. After the selection of the CV to be deleted, we make the user confirm his choice. As a result, the selected CV is deleted from both our portal and the job seeking web site.

COMPANY (EMPLOYEE SEEKER) ROLE



An employee seeker (company) can create a membership, update or delete their membership, send a job announcement and eliminate the CVs that will be conducted to them according to some criteria they give. The effects of these operations are on job seeking web sites. All these operations, except creating company membership, can be reached only when the company is logged in to our portal by the company account. Now let us define these operations that can be done by a company user.

1) Create Company Membership :

When a company wants to use the advantages of our portal, we require them to have an account in our portal. To create the required account, the companies have to fill company membership creation form. The rest of the process is similar to a personal membership creation. The information that is gathered by the form is distributed to job seeking web sites, keeping a copy in our database.

2) Update Company Membership Account :

Some information like telephone number, address or similar information may be changed after the company account has been created. So these information needs to be updated. In order to accomplish this goal, we provide the CV updating service in which we provide the company users a prefilled form. The changes are done on this form. When this form is submitted the updated information is sent to job seeking web sites. We also update our own database.

3) Delete Membership :

A company may be unsatisfied with our services and no longer want to be our user. At these situations, a company may delete their membership. When a company is logged in and want to delete company membership, we direct the user into a confirmation process. At the end of this process, all the account we have created during the creation of company membership is deleted.

4) Send Job Announcement :

Job seeking web sites accept job applications by an amount of payment. This payment avoids companies to leave job applications into a wide range of job seeking web sites. We are planning to make some agreements during the starting period of the collaboration with job seeking web sites. According to these agreements, users will be able to leave job applications with cheaper prices. The cheapness will be determined by us according to the company information and the number of applications they left by using our portal. While leaving a job application, the users select the job seeking web sites they want to leave application. They also give the job specifications. After a payment approval process, the application is distributed to the preferred web sites keeping a copy in our database.

5) Delete Job Announcement :

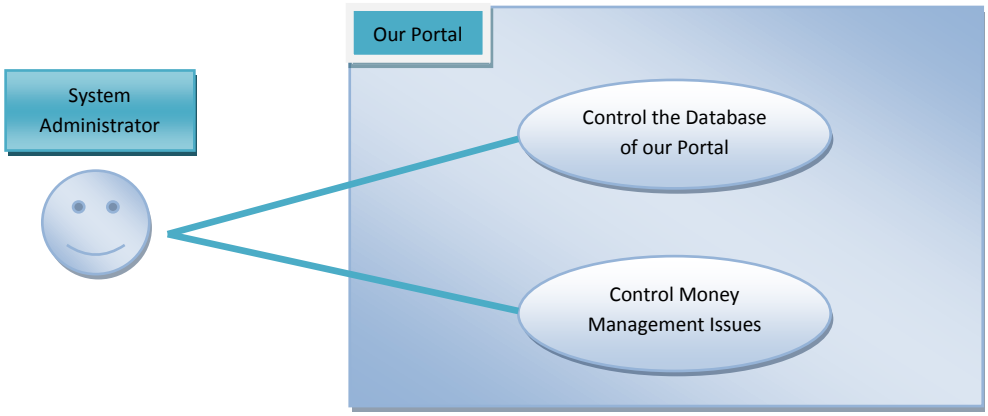
The job applications that are left by a company may be deleted since the position is closed or there is a change in the decision of the company. This service provides the companies to delete this

type of applications. After selecting the job application to be deleted from our portal, the application is deleted from both our database and job seeking web sites database. If this deletion is a result of match, the match is included to statistical information table.

6) CV Elimination Service:

We are planning to provide some CVs according to some match criteria's to companies. These criteria's are defined by the companies. According to these criteria's, the most matching CVs will be send to companies to let them give it a look.

SYSTEM ADMINISTRATOR ROLE



The system administrator of our system has two duties. One is the control of the database of our portal and the other is control of money management issues. Now we want to give these operations a closer look.

1) Control the Database of our Portal :

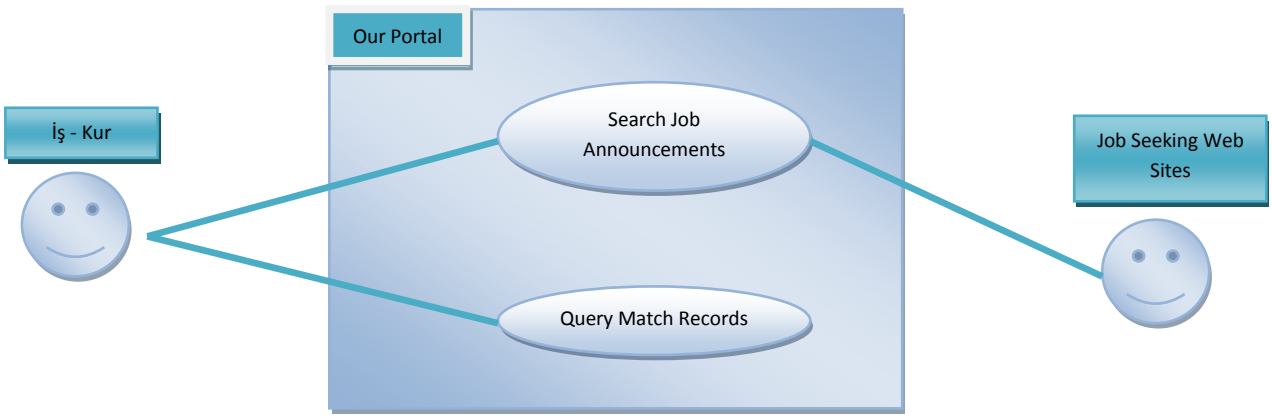
There may be some irrelevant data in our database as a result of misusage of our system. Some users may create accounts with wrong data. There can be numerous problems and wrong data as a result of these types of mistakes. The system administrator works as a clean-up mechanism for this

type of problems. Controlling and cleaning wrong data is one of the main tasks of the system administrator.

2) Control Money Management Issues :

For companies to leave job applications, they have to pay some money to job seeking web sites or our portal. Money management is a difficult issue. So the control of this type of money management issues is the other task of the system administrator.

iş-Kur Role



The services we provide for İş-kur can be summarized by two operations.

1) Search Job Announcements:

İş-kur may want to see the current situation about the employee needs of industries. For this aim, İş-kur may want to make job announcement searches and determine with sector needs more employees and which needs less. For this aim, we provide a job announcement search mechanism for İş-kur. By giving some criteria, they will be able to see the current situation.

2) Query Match Records:

For getting some statistical information about the previous placements of employees, İş-kur may want to get some information about the matches that has been made during a past period of time. This information would provide information about the reduction in the needs of the sectors after a passed time. This information helps İş-kur to see the results of the strategy that is followed for employment control.

ROLE OF JSWSs

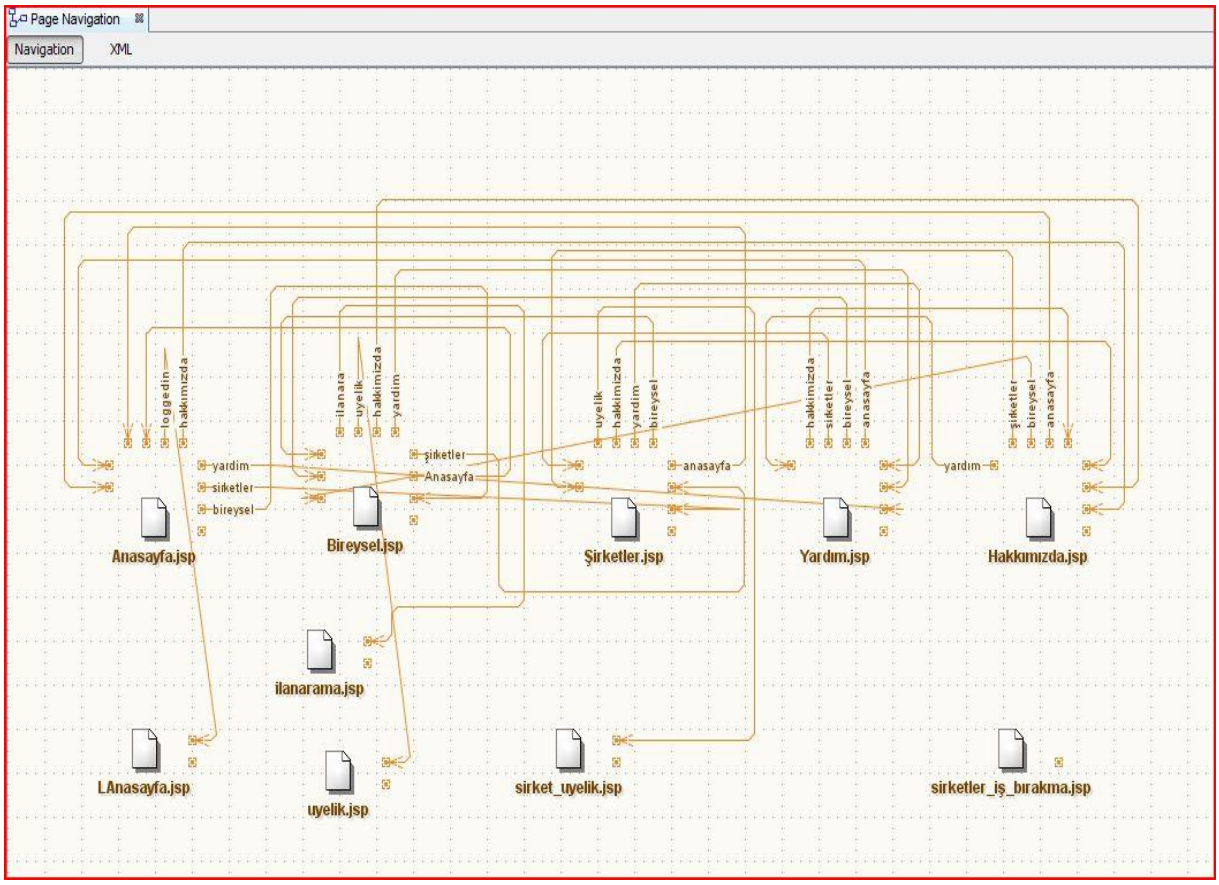
As a final operation, a job seeking web site may declare an unmatched job and want us help in finding a matching CV for the job. This operation is used when there is an announcement for which any qualified CVs cannot be found for a long time. We can think this operation as an emergency trigger of the job seeking web sites. When a request comes the our portal from one of the job seeking web sites, according to the criteria defined, we make CV searches from different job seeking web sites and return the matching results with reference to the communication ways to the owner of that CV.

These are all the main operations we provide to the users of our system. They cover a wide range of properties and they make job seeking easier at the end of the development process of our system.

PART IV – USER INTERFACE DESIGN

When it is time for us to explain the structure of our user interface in detail, we changed it a lot compared to the first design. At the first design, we had used HTML as our language and Dreamweaver as tools of development. But for this design, we used Java as our language and Netbeans Visual Web Pack as our tool development IDE. The reason to select Netbeans is the easiness of implementing web services inside the java codes. We used Visual Web Pack as web page design and creation tool. Now the modification of the web services inside user interface became much more easy.

By means of navigation page we can see the general look of our pages all together. If we look at the navigation page we can see the hierarachy between pages. We want to mention that this diagram doesn't include all the codes that will be used. It only includes the ones we have implemented up to now.



One of the reason for us to select Netbeans as a design tool is the navigation system which can be seen above. By this system, we can relate pages with very basic drag and drop operations. After we create our web pages, we connected them into each other with this navigation page.

If we look into our main page, it consists of the general structure of our portal. The name of our portal is “YENİLİNK”. This is not the completed version. We will develop it in time. As it is shown in the figure below, there are five tabs in our main page. First one takes the users to the homepage of the portal. Second one provides the services for the personal users. Third one is for company services. Fourth one is for help option. We provide help for users who cannot adapt to our portal. Last tab is the “About Us” page. In this main page, there is a login part. By using this part both personal users, companies and administrators can login the system.

The page we have prepared for personal users is “bireysel.jsp” . Its relations with other web pages of our portal can be seen from the navigation tab of Netbeans, which can be seen above. As a personal user, operations can be done which includes creating new membership and searching different job announcements from different job seeking web sites.

When you click the button "Üyelik Oluşturmak İçin", it will direct you to the page, "personal.jsp" to fullfill the needed information for membership. If a person logins as a personal user using his account, the page that he will be directed will be different than the page above because the operations that he can do as member differentiates.

When the user click “Şirketler” tab, the page that comes is “Şirketler.jsp”. If the company users are not logged in to the portal with their account, they cannot do anything from our portal. The services that can be used from our portal are reachable from this tab when the user is logged in. We provide a link for login process and direct the user, if the user is not logged in.

To become a member as a company, you have to fullfill the company membership form. The form can be seen below. After filling the form and pressing the button “Kaydet”, it will go and save the filled information to the required places in the database of our portal and job seeking web sites.

To conclude, the interface of our portal has not been completed. This design is the one that we prepared by intending for web services implementation. As our first web page is designed by using HTML codes, we had to change our design based on java codes. We are aware of the deficiencies in the design. For example, we have not prepared ccs(cascading style sheet) pages yet. We continue our design step by step.

PART V - PROCESS SPECIFICATIONS & WEB SERVICES DESIGN

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

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.

5.1.1. Web Services of the JSWSs

Job Seeker Membership Acceptance Service

- **bool** usernameControl(**String** name)
- **bool** insertJobSeekerMembershipInfo(**String** [] info)

Job Seeker Membership Updater Service

- **bool** changeJobSeekerMembershipInfo (**String** username, **String** [] newInfo)

Job Seeker Membership Deletion Service

- **bool** deleteJobSeeker(**String** username)

CV Acceptance Service

- **bool** insertNewCV(**String** username, **String** [] cvInfo)

CV Updater Service

- **bool** replaceCVWithNewCV1(**String** username, **String** cvName, **String []** newCVInfo)
- **bool** replaceCVWithNewCV2(**int** cvId , **String []** newCVInfo)

CV Deletion Service

- **bool** deleteCV1(**String** username, **String** cvName)
- **bool** deleteCV2(**int** cvID)

CV Search Service

- **int** [] returnCVsWithTheseCriteria(**String** [] criteria)
- **int** returnCVId(**String** username , **String** cvName)
- **String**[] returnCVOwnerAndTitle (**int** cvId)
- **String** [] returnDetailedCVInfo1(**int** cvId)
- **String** [] returnDetailedCVInfo2(**String** username , **String** cvName)

Company Membership Acceptance Service

- **bool** usernameControl(**String** name)
- **bool** insertCompanyMembershipInfo (**String** [] info)

Company Membership Updater Service

- **bool** changeCompanyMembershipInfo(**String** username, **String** [] info)

Company Membership Deletion Service

- **bool** deleteCompany(**String** username)

Job Announcement Acceptance Service

- **bool** insertNewAnnouncement(**String** username, **String** [] jobAnnInfo)
- **int** [] getAddressIdsOfTheCompany(**String** username)
- **String** [] getAddress(**int** addressId)

Job Announcement Deletion Service

- **bool** deleteJobAnnouncement1(**String** username, **String** jobAnnName)
- **bool** deleteJobAnnouncement2(**int** jobAnnId)

Job Announcement Search Service

- **int [] returnJobAnnsWithTheseCriteria(String [] criteria)**
- **int** returnJobAnnId(**String** username , **String** jobAnnName)
- **String []** returnJobAnnOwnerCompanyAndTitle(**int** jobAnnId)
- **String [] returnDetailedJobAnnInfo1(int jobAnnId)**
- **String [] returnDetailedJobAnnInfo2(String username , String jobAnnName)**

Job Announcement Control Service

- **int** [] returnCVIdsAppliedToJobAnn1(**int** jobAnnId)
- **int** [] returnCVIdsAppliedToJobAnn2(**String** username, **String** jobAnnName)
- **bool** deleteCVfromJobAnnList(**int** jobAnnId , **int** cvId)
- **bool** approveCVInJobAnnList(**int** jobAnnId , **int** cvId)

CV Job Match Information Service
- **int** [] returnLastMatches(**Date** dateAfter)
- **String** [] returnDetailedMatchInfo (**int** matchId)

5.1.2. Web Services of our Portal

Money Deposit Service
- **bool** addMoneyToCompanyAccount(**String** username , **double** amountOfMoney)

Unmatched Job Declaration Service
- **String** [] findCVsforMyJobAnn1(**String** jswsName , **int** unappliedJobAnnId)
- **String**[] findCVsforMyJobAnn2(**String** jswsName , **String** [] jobAnnCriteria)

5.1.3 Description of Web Services provided by JSWSs

1) Job Seeker Membership Acceptance Service: This service will give us the chance to create a new membership on the JSWS for the job seeker who uses our portal. When the job seeker creates a membership in our portal, we will use this service of JSWSs to create a membership for him/her with different username and password about which only our portal knows. Our portal's database holds all membership information of its members in JSWSs as well as the membership information of them in our portal. This service consists of two methods.

bool usernameControl(**String** name)

We are going to use this method to check whether the produced username for the job seeker exists or not. As a result of this check, the method will return us a boolean value. Depending on this

return value, our portal will produce a new username for the job seeker and will check its availability again until it finds a unique username.

bool insertJobSeekerMembershipInfo (**String** [] info)

After all the fields are filled by the user in the registration step and "Gönder" button is pressed, our portal will store this information in its database, and will add records to EmployeeMembership_JSWS_Queue and trigger a module to send unsent membership information of job seekers whose references are in that queue table. If a JSWS under our portal is not available at that time, it will not be a problem, i.e., that job seekers membership information will not be lost because by the means of this queue table, we know which membership information could not be sent. In addition to this queue table, there will be a timer module in our system which periodically triggers a module to send unsent membership information.

"info" parameter will be a string array which has five elements, namely, username, password, secret question, answer, and email. Only email will be the same with the job seeker's real email given to our portal when registering. Since the other fields are known only by our portal, they are not necessarily the same with the information given by the job seeker to our portal when registering.

2) Job Seeker Membership Updater Service: This service will be used when a job seeker updates his/her membership information in our portal. After we make the necessary updates in our portal's database, we will add records to Updated_ JobSeekerMembership _Queue and trigger a module to send updated informations to JSWSs. This queue table, like the previous one solves the problem of loosing data because of unavailability of a JSWS.

bool changeJobSeekerMembershipInfo (**String** username, **String** [] newInfo)

"newInfo" parameter is a string array which has only one element, namely, email. Since only email in JSWS database will be the real email of the job seeker given to our portal, it is sufficient to update only the email field of the job seeker membership information in JSWSs.

3) Job Seeker Membership Deletion Service: This method will be used by our portal for deleting a member from JSWSs. When a job seeker wants to delete his/her membership from our portal, we first invoke this web service and after that we will also delete the membership of him/her from our portal's database.

bool deleteJobSeeker (**String** username)

Since "username" is unique for all job seekers, we will use this field for specifying the person whose membership will be deleted.

4) 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 will store that CV to our portal's database and add records to a queue table, namely CV_JSWS_Queue. Like the previous ones, by triggering a module of our system, we will send the unsent CVs which has references in queue table to JSWSs.

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 will be sent to JSWSs, and this method will return a boolean value to our portal about the successfulness of the insertion operation. If insertion failed, we will not delete the related record in the queue table of our portal's database to be able to send the CV again in the following triggering.

5) CV Updater Service: A job seeker can edit his/her CVs whenever he/she wants in our portal. If he/she makes an update on his/her CV, then we will update the necessary tables of our portal's database and add records to Updated_CV_JSWS_Queue to make the necessary updates in all JSWSs without losing any information. Triggering events will also occur here, like the previous ones.

bool replaceCVWithNewCV1(**String** username, **String** cvName, **String []** newCVInfo)

bool replaceCVWithNewCV2(**int** cvId , **String []** newCVInfo)

Since “job seeker username” and “cvName” couple and “cvId” uniquely specifies a CV in a JSWS, there are 2 methods which can be used with the same purpose.

6) CV Deletion Service: Since a job seeker can create a lot of CVs for him/her, he/she may want to delete some of them. When he/she pushes the delete button by selecting a CV in CVs page of the user interface, we will delete that CV in our portal’s database and by the means of this web service, we will also delete the corresponding CV in all JSWSs. The methods of this web service are the following ones:

bool deleteCV1(**String** username, **String** cvName)

bool deleteCV2(**int** cvID)

7) CV Search Service: If a job announcement is not applied by 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’ databases for finding CV matches with that unapplied job announcement. In order to accomplish this task we require JSWSs to provide us a CV Search Web Service. This service consists of the following 5 methods:

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

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

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

String [] returnDetailedCVInfo1(**int** cvId)

String [] returnDetailedCVInfo2(**String** username , **String** cvName)

With those methods, our portal can collect the CV ids which meet some criteria and CV details can also be taken.

8) 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 produced user name for the company exists or not in the JSWS which the web service belongs to. Depending on the value returned, our portal can produce a new username for the company, and test it with this method again.

bool insertCompanyMembershipInfo (**string** [] info)

The company membership information is sent as a string array with this method and the company account is to be created in JSWS. If the account cannot be created, false is returned, its record in the CompanyMembership_JSWS_Queue will not be deleted to be able to send this membership information in the next trial.

9) Company Membership Updater Service: There are many things that can change in a company. For example, employee number of the company, telephone number of the responsible person, etc … The responsible person of the company can reflect those changes to our portal by changing the membership information of its company. In that case, our portal must reflect those changes to all JSWSs. Therefore, we require a company membership update service from JSWSs.

bool changeCompanyMembershipInfo(**String** username, **String** [] info)

With the "info" parameter, we will send all up-to-date membership information of the company to JSWS which this web service belongs to.

10) Company Membership Deletion Service: By the means of this web service, our portal can delete the company from JSWS's databases which deletes its accounts from our portal.

bool deleteCompany(**string** username)

The "username" parameter is the company username which uniquely specifies the company.

11) Job Announcement Acceptance Service: In order to insert job application to JSWSs, we use this web service. When a company selects a JSWS to publish its job announcement, this service is invoked by our portal and that job announcement will be sent to the selected JSWS. In order to publish a job announcement, the company must have enough money in our portal's account in the bank. If not, the company can deposit money to our portal's account; in that case, the bank will inform us about the money deposit to our account with the web service of our portal, namely Money Deposit Service. For every job announcement publishing in a JSWS, that JSWS will update the "receivableMoneyAmount" field in its "WebServiceUsers" table to be able to know how much money that our portal must give to it because of publishing of a job announcement. The same information is also stored in our portal's database, i.e., for every job announcement publishing in a JSWS through our portal, we will update the "deptAmount" field of the "JSWSs" table so that our portal is able to know how much money it must give to each JSWS.

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

If the conditions mentioned in the method above are satisfied, the job announcement details will be sent as a string array. The job announcement will be published in the selected job seeking web sites.

int [] getAddressIdsOfTheCompany(**String** username)
String [] getAddress(**int** addressId)

In the job announcement creation page, all addresses of the company will be shown with the radio buttons, so that one of them can be selected as a work place of the candidate employee. Therefore, the two methods above are added to this web service.

12) Job Announcement Deletion Service: When a company publish a job announcement in a JSWS through our portal and finds an employee, then that company may want to drop its job announcement. In that case, the responsible person of the company will sign in to our portal, and open its job announcements page and delete one of the job announcement of his/her company. Our portal deletes that job announcement in its database and deletes the corresponding job announcements in JSWSs by the means of this web service.

bool deleteJobAnnouncement1(**String** username, **String** jobAnnName)

bool deleteJobAnnouncement2(**int** jobAnnId)

Since "company username" and "job announcement name" uniquely identifies a job announcement as well as "job announcement id", both methods can be used for deletion.

13) 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. Our portal requires this web service from all JSWSs to accomplish this search. 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. With that integer array, we will take the necessary fields of job announcements to show the result in our web page.

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

This method is to be used for finding the corresponding job announcement in a JSWS which is also in our portal's database since the same job announcement has two different ids in different databases, but the job announcement name is the same for both of them, we can find its id in JSWS by the means of this method.

String [] returnJobAnnOwnerCompanyAndTitle(**int** jobAnnId)

After the returnJobAnnsWithTheseCriteria method is called and the job announcements' ids which are meeting the given criteria are taken from a JSWS, we can find those job announcements' titles and the company usernames by the means of this method.

String [] returnDetailedJobAnnInfo1(**int** jobAnnId)

This method will be used by our portal 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.

String [] returnDetailedJobAnnInfo2(**String** username , **String** jobAnnName)

Since "company username" and "job announcement name" uniquely identifies a job announcement as well as "job announcement id", this method also can be used by our portal.

14) Job Announcement Control Service: After a company leaves a job announcement to a JSWS through our portal, and job seekers apply this job announcement by leaving their CVs to it, then company need to manipulate those CVs, i.e., a company want to delete a CV from the list of applied CVs to its job announcement because qualifications in a CV may be not sufficient and that CV is to be dropped the list. Besides when a company likes the qualification in a CV by looking the details of that CV, and after an interview with the job seeker which that CV belongs to, that job seeker can be hired by the company. In that case, the company will approve that CV from its list and this CV-job announcement matching will be added to the Job CV Match Archive tables in JSWSs whose last records will be taken by our portal in periodically. This web service has the following methods:

int [] returnCVIdsAppliedToJobAnn1(**int** jobAnnId)

int [] returnCVIdsAppliedToJobAnn2(**String** username, **String** jobAnnName)

bool deleteCVfromJobAnnList(**int** jobAnnId , **int** cvId)

bool approveCVInJobAnnList(**int** jobAnnId , **int** cvId)

15) CV Job Match Information Service: When a company approves a CV which is left to its job announcement, this match will be added to Job CV Match Archive tables in JSWSs. The last records in those tables will be taken by our portal in periodically, and will be added to "JobCVMatchArchive" table in our portal's database. When İş-Kur signs in to our portal and queries our portal with some

criteria, we search in the “JobCVMatchArchive” table according to those criteria and return a statistical information with the graphs showing those statistics in an understandable manner. This web service has 2 methods:

int [] returnLastMatches(**Date** dateAfter)

String [] returnDetailedMatchInfo (**int** matchId)

5.1.4. Description of Web Services provided by our Portal

1) Money Deposit Service: When a company wants to publish a job announcement in a JSWS through our portal, it is required that company deposited sufficient amount of money to portal’s account in the bank. If there is a deposit to our portal’s account, the bank will use this web service of our portal to inform us about it. There is only one method of this web service:

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

When the method above is called by the bank, we add the amount of money to the “CashInTheBox” of the “EmployeeSeekerMembershipInfo” table in our portal’s database. By the means of the first parameter “username”, we know which company deposited money to our account.

2) Unmatched Job Declaration Service: This web service will be used by the JSWSs. They will use this service when no CV is applied to a job announcement for a specific time period. When this service is called, our portal searches other JSWSs’ CV database and tries to find matches for the job announcement. This web service has two methods:

String [] findCVsforMyJobAnn1(**String** jswsName, **int** unappliedJobAnnId)

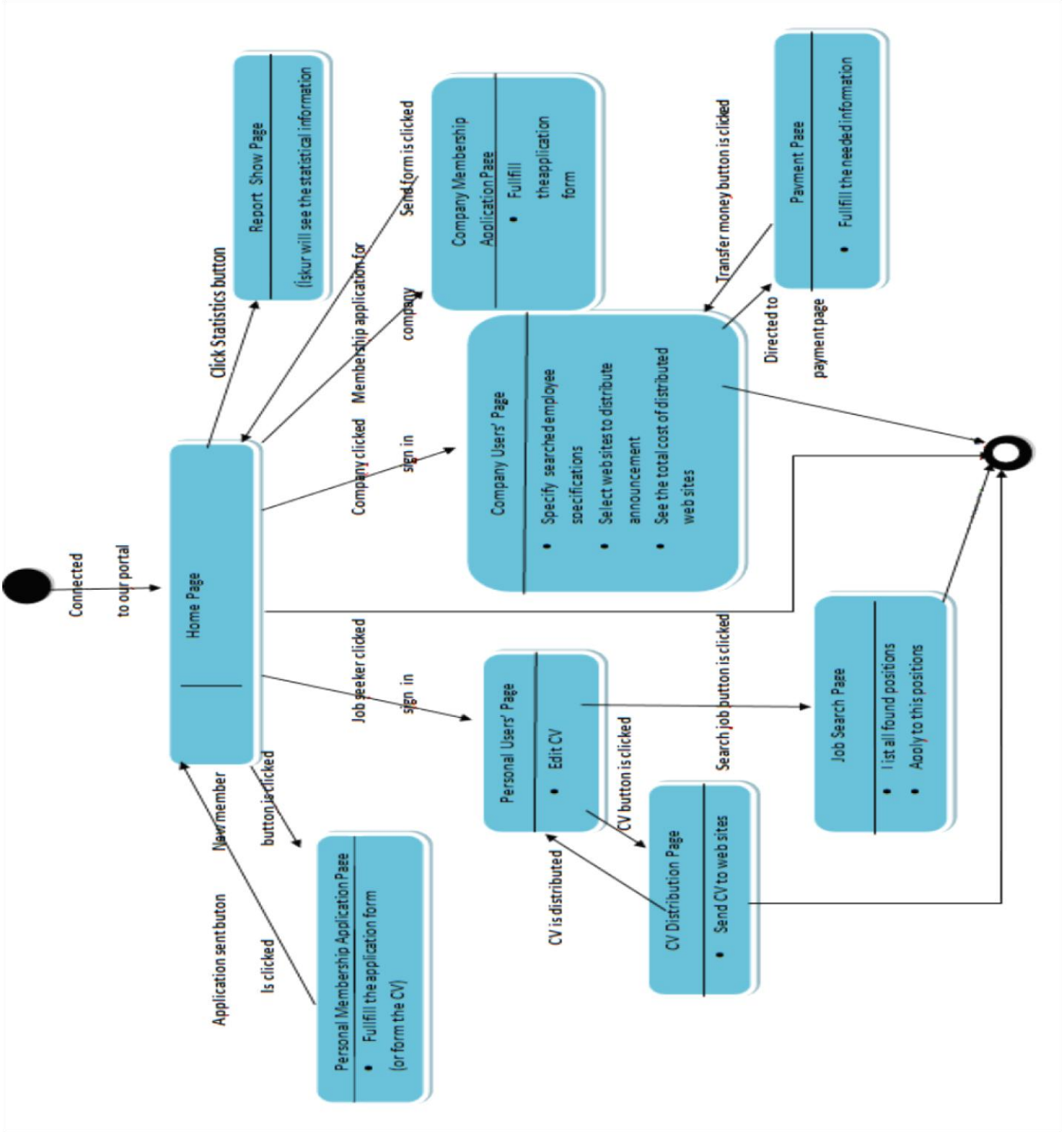
String[] findCVsforMyJobAnn2(**String** jswsName, **String** [] jobAnnCriteria)

The first parameter is the name of the caller JSWS of the methods above. As a second parameter, only the job announcement id can be given or the job announcement's criteria can also be given. For every match, we will return two strings, namely the jsws name whose CV database consists of the matched CV, and the id of that CV.

NOTICE! Employee seekers cannot 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

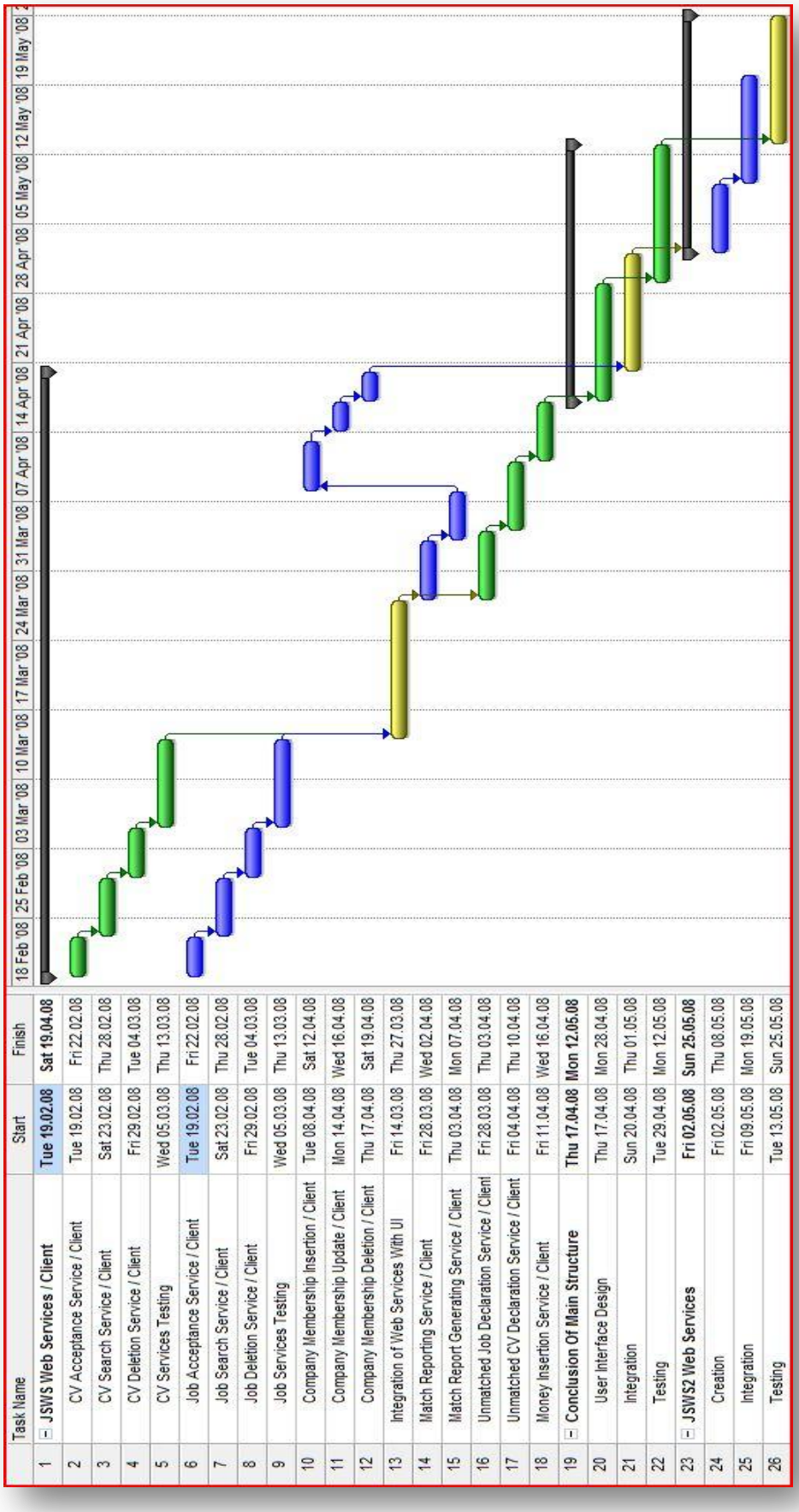
In general this diagram will show you an overall look to the portal we have designed. In our system, through our main web page, we can reach some sub pages or sub states. The state of the system is changed by the user interactions like clicking on buttons. When a button is clicked, a new page will be 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.


Connected
to our portal
Home Page
Click Statistics button
Report Show Page
(Iskur will see the statistical information
Company clicked
sign in
Membership application for
company
Send form is clicked
Company Membership Application Page
Fullfill theapplication form
Company Users' Page
Specify searched employee specifications
Select web sites to distribute announcement
See the total cost of distributed web sites
Transfer money button is clicked
Directed to payment page
Payment Page
Fullfill the needed information
Job seeker clicked
sign in
Personal Users' Page
Edit CV
CV button is clicked
CV is distributed
CV Distribution Page
Send CV to web sites
Search job button is clicked
Job Search Page
List all found positions
Apply to this positions
New member
button is clicked
Application sent buton
Is clicked
Personal Membership Application Page
Fullfill the application form
(or form the CV)

PART VI - WHAT WE HAVE DONE SO FAR

At the beginning of the semester we started our project with a learning period. At this time interval, we analyzed about Web services and user interface design. We started the design of our database; we designed the database of our portal and the other job seeking Web sites in detail. After we get the feedback of initial design report, we made big changes in our database design and we completed the design of our database again. We created three databases. The first one is for our portal and the other two for the job seeking web sites which we will simulate. At the same time, we tried to design the interface of our portal. For the implementation, first of all we wanted to use the tools that SoftwareAG provides to us. Since there are some problems between our PCs and the given tools, we could not use those tools. Then, we started to search some user interface design tools, and we learned that there is a plug-in of Netbeans, namely Netbeans Visual Pack, for the interface design of web sites. We have already learned to create the client sides of web services in Netbeans, so we will use NetBeans for all the implementations we make excluding web services of the server side. In addition, we have also learned to create web services in Eclipse with Axis2 plug-in so we will use Eclipse for the implementation of web services. We have selected PostgreSQL as our database management system. On the other hand, since there were many assignments and exams, we are a bit late to follow the Gantt Charts so it needed to do be updated. Up to now we have been able to learn the tools, finished database design and write a web service, namely “membership acceptance service” with its client. We plan to speed up our progress and continue to the implementation during semester holiday.

Gantt chart for the first semester



	Task Name	Start	Finish
1	JSWS Web Services / Client	Tue 19.02.08	Sat 19.04.08
2	CV Acceptance Service / Client	Tue 19.02.08	Fri 22.02.08
3	CV Search Service / Client	Sat 23.02.08	Thu 28.02.08
4	CV Deletion Service / Client	Fri 29.02.08	Tue 04.03.08
5	CV Services Testing	Wed 05.03.08	Thu 13.03.08
6	Job Acceptance Service / Client	Tue 19.02.08	Fri 22.02.08
7	Job Search Service / Client	Sat 23.02.08	Thu 28.02.08
8	Job Deletion Service / Client	Fri 29.02.08	Tue 04.03.08
9	Job Services Testing	Wed 05.03.08	Thu 13.03.08
10	Company Membership Insertion / Client	Tue 08.04.08	Sat 12.04.08
11	Company Membership Update / Client	Mon 14.04.08	Wed 16.04.08
12	Company Membership Deletion / Client	Thu 17.04.08	Sat 19.04.08
13	Integration of Web Services With UI	Fri 14.03.08	Thu 27.03.08
14	Match Reporting Service / Client	Fri 28.03.08	Wed 02.04.08
15	Match Report Generating Service / Client	Thu 03.04.08	Mon 07.04.08
16	Unmatched Job Declaration Service / Client	Fri 28.03.08	Thu 03.04.08
17	Unmatched CV Declaration Service / Client	Fri 04.04.08	Thu 10.04.08
18	Money Insertion Service / Client	Fri 11.04.08	Wed 16.04.08
19	Conclusion Of Main Structure	Thu 17.04.08	Mon 12.05.08
20	User Interface Design	Thu 17.04.08	Mon 28.04.08
21	Integration	Sun 20.04.08	Thu 01.05.08
22	Testing	Tue 29.04.08	Mon 12.05.08
23	JSWS2 Web Services	Fri 02.05.08	Sun 25.05.08
24	Creation	Fri 02.05.08	Thu 08.05.08
25	Integration	Fri 09.05.08	Mon 19.05.08
26	Testing	Tue 13.05.08	Sun 25.05.08

Gantt chart for the second semester

Appendix A - Job Categorization

We again looked at the job seeking web sites like Kariyer.net, Yenibiris.com, SecretCV.com and created our categorizations according to these sites. The main sectors we could find are as follows:

- **AR-GE/Bilişim/Teknoloji/Internet/Telekomünikasyon**
- **Bakım/Onarım**
- **Basın/Yayın/Medya/Reklamcılık**
- **Beyaz Eşya/mobilya**
- **Denizcilik**
- **Eğitim/Danışmanlık**
- **Eğlence/Restaurant**
- **Finans/Bankacılık**
- **Gıda/Biyoteknoloji/Sağlık/İlaç**
- **Güvenlik/SavunmaSanayi**
- **Halkla ilişkiler/İdari İşler/Sekreterya**
- **Hukuk**
- **İnşaat/GayriMenkul**
- **Konfeksiyon/Giyim/Tekstil**
- **Kozmetik/Kimya/PetroKimya**
- **Lojistik/Dağıtım/Taşımacılık/Ulaştırma**
- **Madencilik**
- **Mühendislik/Mimarlık**
- **Otomotiv**
- **Pazarlama/Muhasebe/Satış ve Satın Alma/Dokümantasyon**
- **Sigortacılık**
- **Spor**
- **Turizm/Konaklama**
- **Üretim/İmalat**
- **Ziraat/Tarım/Hayvancılık**

The jobs are grouped under these sectors. In the rest of the report, we will show these job and sector groupings.

AR-GE / Bilişim / Teknoloji / Internet / Telekomünikasyon

- **AR-GE Müdürü/Uzmani**
- **AR-GE Mühendisi**
- **Bilgi İşlem Yöneticisi**
- **Bilgi Sistemleri Yöneticisi**
- **Bilgisayar Mühendisi**
- **Bilgisayar Operatörü**
- **Bilgisayar Programcısı/Analist**
- **Donanım Mühendisi**
- **Elektrik Mühendisi**
- **Elektrik/Elektronik Mühendisi**
- **Elektronik Mühendisi**
- **ERP Uzmanı**
- **Internet Hizmetleri Yöneticisi**
- **Kalite Kontrol Mühendisliği**
- **MIS Direktörü**
- **MIS Uzmanı**
- **MRP Uzmanı**
- **Multimedya Tasarımcısı**
- **Network Mühendisi**
- **Network Yöneticisi**
- **Saha Mühendisi**
- **Sistem Analisti**
- **Sistem Mühendisi**
- **Sistem Yöneticisi**
- **Tasarım Mühendisi**
- **Teknik Destek Müdürü**
- **Teknik Destek Müşteri Temsilcisi**
- **Teknik Destek Uzmanı**
- **Telekomunikasyon Mühendisi**
- **Uygulama Mühendisi**
- **Veritabanı Uzmanı**
- **Veritabanı Yöneticisi/Müdürü**
- **Web/Internet Uzmanı/Tasarımcısı**
- **Yazılım Geliştirme Uzmanı**
- **Yazılım Mühendisi**
- **Yazılım Tasarım Mühendisi**
- **--- Diğer ---**

Bakım/Onarım

- **Bilgisayar Bakım ve Onarımcısı**
- **Bilgisayar Donanım Teknikeri/Teknisyeni**
- **Doğalgaz ve Isıtma ve Sıhhi Tesisat Teknikeri/Teknisyeni**
- **İklimlendirme ve Soğutma Teknikeri**
- **İş Makineleri Tamircisi**
- **Kaporta Tamircisi**
- **Karo-Fayans Döşeyicisi**
- **Kaynakçı**
- **Konfeksiyon Makineleri Bakım ve Onarım Teknisyeni**
- **Saat Tamircisi**
- **--- diger bakım ve onarım pozisyonları ---**

Basın/Yayın/Medya/Reklamcılık

- **Editör**
- **Editör Yardımcısı**
- **Fotoğrafçı**
- **Gazete muhabiri**
- **Kameraman**
- **Radyo Televizyon Muhabiri**
- **Reklam Müdürü/Şefi**
- **Reklam Uzmanı**
- **Spiker**
- **Teknik Editör**
- **Tiyatro Oyuncusu**
- **Yapımcı**
- **--- Diğer Pozisyonlar ---**

Beyaz Eşya/mobilya

- **Ağaç Oymacısı**
- **--- Diğer ---**

Denizcilik

- **armatör**
- **Deniz ve Liman İşletme Meslek Elemanı**
- **Gemi Elektroniği ve Haberleşme Teknisyeni**
- **Gemi İnşa teknisyeni**
- **Gemi İnşaatı Mühendisi**
- **Gemi Kaptanı**
- **Gemi Makineleri Mühendisi**
- **Gemi ve Deniz Yapıları Mühendisi**
- **Güverte Teknikeri**
- **--- Diğer ---**

Eğitim/Danışmanlık

- **arşivci**
- **Bilgi İşlem Sistemleri Danışmanı**
- **Eğitim Danışmanı**
- **ERP Danışmanı**
- **Hukuk Danışmanı**
- **İnsan Kaynakları Danışmanı**
- **Mali İşler Danışmanı**
- **MRP Danışmanı**
- **Öğrenim Danışmanı**
- **Öğretmen**
- **Personel Seçme Yerleştirme Danışmanı**
- **Teknik Danışman**
- **Vergi Danışmanı**
- **Yönetim Danışmanı**
- **--- Diğer Danışmanlık Pozisyonları ---**

Eğlence/Restaurant

- **Aşçı**
- **Garson**
- **Komi**
- **--- Diğer ---**

Finans/Bankacılık

- **Banka İhracat/İthalat Müdürü/Şefi**
- **Banka İhracat/İthalat Uzmanı**
- **Banka Memuru**
- **Banka Müfettişi**
- **Banka Operasyon Müdürü**
- **Banka Operasyon Şefi**
- **Borsa Uzmanı**
- **Bütçe Raporlama Müdürü**
- **Bütçe Raporlama Uzmanı**
- **Finans Kontrolörü**
- **Finansal Planlama Uzmanı**
- **Finansman Şefi**
- **Finansman/Mali İşler Direktörü**
- **Finansman/Mali İşler Müdürü**
- **Fon Yöneticisi**
- **Hazine Müdürü/Şefi**
- **Hazine Uzmanı**
- **İstatistikçi**
- **İşletmeci**
- **Kredi Pazarlama Müdürü**
- **Kredi Pazarlama Uzmanı**
- **Mali İşler Analisti**
- **Pazarlama Yönetmeni**
- **Sigorta Brokeri**
- **Şube Müdür Yardımcısı**
- **Şube Müdürü**
- **Teknik Destek Müşteri Temsilcisi**
- **Teknik Destek Uzmanı**
- **Veznedar**
- **--- Diğer Bankacılık/Finans Pozisyonları ---**

Gıda/Biyoteknoloji/Sağlık/İlaç

- **Ambulans ve Acil Bakım Teknikeri**
- **Anestezi Teknikeri**
- **Cerrah**
- **Doktor**
- **Ebe**
- **Eczacı**
- **Farmakolog**
- **Fırıncı**
- **Fizyoterapist**
- **Gıda Teknikeri**
- **Güzellik Uzmanı**
- **Hemşire**
- **Laborant**
- **Pratisyen Hekim**
- **Protez ve Ortez Teknikeri**
- **Psikolog**
- **Radyoloji (Röntgen) Teknisyeni**
- **Sağlık Memuru**
- **Veteriner Hekim**
- **--- Diğer ---**

Güvenlik/Savunma Sanayi

- **gardiyan**
- **güvenlik görevlisi**
- **İtfaiyeci**
- **Komiser**
- **Polis Memuru**
- **--- Diğer ---**

Halkla ilişkiler/İdari İşler/Sekreterya

- **Halkla İlişkiler ve Tanıtım Elemanı**
- **İnsan Kaynakları Müdürü**
- **Sekreter**
- **--- Diğer ---**

Hukuk

- **avukat**
- **hakim**
- **hukuk sekreteri**
- **savcı**
- **--- diğer hukuk pozisyonları ---**

İnşaat/GayriMenkul

- **Amele**
- **Duvarcı**
- **İnşaat Teknisyeni**
- **Sıvacı**
- **Şantiye Şefi**
- **--- Diğer ---**

Konfeksiyon/Giyim/Tekstil

- **İplik Teknisyeni**
- **Moda Tasarımı Öğretmeni**
- **--- Diğer ---**

Kozmetik/Kimya/PetroKimya

- **Kimyager**
- **kuru temizlemeci**
- **Rafineri ve Petrokimya Teknikeri**
- **Petro-Kimya Teknisyeni**
- **--- Diğer ---**

Lojistik/Dağıtım/Taşımacılık/Ulaştırma

- **Bilet Satış Elemanı**
- **Hostes**
- **Kabin Memuru**
- **Pilot**
- **Yer Hostesi**
- **Uçak Motorları Bakım Onarım Teknisyeni**
- **--- diger ---**

Madencilik

- **Maden Teknisyeni**
- **--- diger ---**

Mühendislik/Mimarlık

- **Ar-Ge Mühendisliği**
- **Bilgisayar Mühendisi**
- **Biyomedikal Mühendisi**
- **Çevre Mühendisi**
- **Deniz Teknolojileri Mühendisi**
- **Deniz Ulaştırma İşletme Mühendisi**
- **Elektrik Mühendisi**
- **Elektrik - Elektronik Mühendisi**
- **Elektronik Mühendisi**
- **Elektronik ve Haberleşme Mühendisi**
- **Endüstri Mühendisi**
- **Fizik Mühendisi**
- **Gemi İnşaatı Mühendisi**
- **Gemi Makineleri Mühendisi**
- **Gıda Mühendisi**
- **Havacılık / Uçak Mühendisi**
- **Hidrojeoloji Mühendisi**
- **İç Mekan Tasarımı Teknisyeni**
- **İç Mimar**
- **İnşaat Mühendisi**
- **İşletme Mühendisi**
- **Jeodezi ve Fotogrametri Mühendisi**
- **Jeofizik Mühendisi**
- **Jeoloji Mühendisi**
- **Kalite Güvence Mühendisliği**
- **Kalite Kontrol Mühendisliği**
- **Kimya Mühendisi**
- **Lojistik Mühendisi**
- **Maden Mühendisi**
- **Makine Mühendisi**
- **Matematik Mühendisi**
- **Metalürji ve Malzeme Mühendisi**
- **Meteoroloji Mühendisi**
- **Nükleer Enerji Mühendisi**
- **Orman Mühendisi**
- **Otomotiv Mühendisi**
- **Petrol ve Doğal Gaz Mühendisi**
- **Sistem Mühendisi**
- **Tekstil Mühendisi**
- **Telekomünikasyon Mühendisi**
- **Üretim Mühendisi**
- **Uzay Mühendisi**
- **Yazılım Mühendisi**
- **Ziraat Mühendisi**
- **---Diğer Mühendislik Pozisyonları---**

Otomotiv

- **Otomotiv Teknikeri**
- **--- diger ---**

Pazarlama/Muhasebe/Satış ve Satın Alma/ Dokümantasyon

- **arşivci**
- **Kasiyer**
- **--- diger ---**

Sigortacılık

- **Genel Sigortacılık Uzmanı**
- **Hayat Sigortası Uzmanı**
- **Sağlık Sigortası Uzmanı**
- **Satış Temsilcisi**
- **Sigortacılık Meslek Elemanı**
- **---Diğer Sigortacılık Pozisyonları---**

Spor

- **Başantrenör**
- **Antrenör**
- **Atletik Antrenör/Kondisyoner**
- **Fizyoterapist**
- **Hakem**
- **İstatistik Antrenörü/Analist**
- **Masör**
- **Yardımcı Antrenör**
- **---Diğer Spor Pozisyonları---**

Turizm/Konaklama

- **Animatör**
- **Ahçı**
- **Barmen/Barmaid**
- **Otel Müdürü**
- **Ön Büro Elemanı**
- **Ön Büro Müdürü/Şefi**
- **Restoran Müdürü**
- **Rezervasyon Müdürü/Şefi**
- **Şef Ahçı**
- **Tur Operatörü**
- **Turist Rehberi**
- **Yiyecek/İçecek Müdürü/Şefi**
- **Ziyafet Koordinatörü**
- **--- Diğer ---**

Üretim/İmalat

- **--- tum üretim pozisyonları ---**

Ziraat/Tarım/Hayvancılık

- **Arıcılık Teknikeri**
- **Bahçeler Direktörü**
- **besicilik teknikeri**
- **Bitki Koruma Direktörü**
- **Fidan Üretim ve pazarlama Müdürü**
- **Fidan Yetiştirme Teknikeri**
- **Hayvan Yetiştiriciliği ve Sağlığı Teknikeri**
- **Peyzaj Mühendisi**
- **Peyzaj Uygulama ve Süs Bitkileri Teknikeri**
- **Plantasyon Sorumlusu**
- **Sebze Üretim Teknikeri**
- **Sulama Projelendirme Sorumluları**
- **Süt ve Ürünleri Teknikeri**
- **Tarım Alet ve Makineleri Teknikeri**
- **ziraat mühendisi**
- **--- Diğer ---**