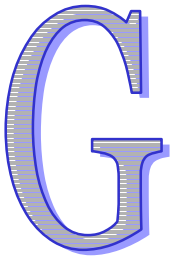
BLUEPRINT Software
a development company

# GUIDS Document

***For a Virtual Placement Agency***

BluePrint Team
Pheonia Chang
Thomas Lai
Phoebe Lam
Natalie McGee
Karim Nanji
Martin Palmer

# Table of Contents



# 1. Application Goals and Objectives

## *Main Purpose*

The goal of this project is to develop an enterprise application that will allow the recruiting staff at Virtual Placement Agency (VPA) to manipulate information on job openings and placements managed by the agency. This will require the tracking of job postings, hiring companies, and applicants.

## *Business Objectives*

- To track and maintain the three major components of the business : Companies, Applicants, Positions.
- Match applicants to available positions.
- Allow applicants looking for employment to access information on available job openings by accessing the VPA's website.

## *Technical Objectives*

- Formalize the design of the software using UML modeling techniques and the GUIDS methodology.
- Follow and object-oriented design methodology to ensure the application is properly defined before it is built.
- Use Microsoft Windows GUI Standards to develop a user-friendly interface.
- Provide a software environment that operates effectively with Microsoft Windows NT Server, Windows 98 and Access 97.
- Design scalable application that can be expanded as the agency grows.
- Develop a distributed component-based system that runs on a client/server network .
- Develop Active X components, using Visual Basic 6, that are managed by using Microsoft Transaction Server (MTS).
- Creation of the web enabled application using Active Server Pages (ASP).

# 2. Use Cases

The following use cases outline the typical business processes the Placement Application will need to perform. These will be used to validate design decisions taken throughout system development.

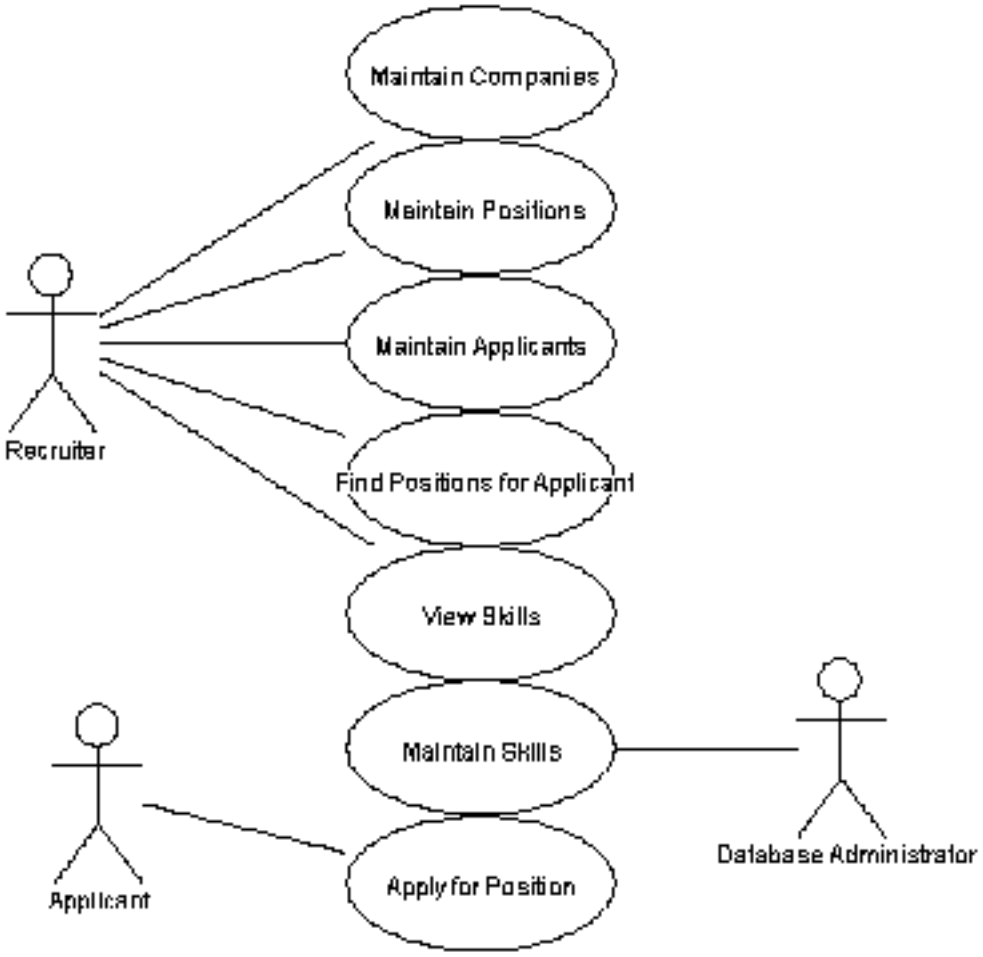



Figure 1—Use Case Diagram for Placement Application

## *Use Case—Maintain Companies*

### Section—Main

Actors— Recruiters

Purpose—Allows the agency to keep profiles on their client companies for current and future business.

Overview—The agency keeps a list of client companies with staffing needs that fit the high-tech niche of the agency. When a company calls the agency for the first time to post a position, that company will be added to the list. Sometimes the agency will add other known high-tech companies that could benefit from the services provided by the agency (even if they have yet to do any business with the agency).

Sometimes company profiles change, such as when the company relocates, changes its name or phone number. The system will allow the modification of company profiles to keep them up to date.

Companies are not to be deleted from the system.

### Typical Course of Events

1. This use case begins when a Recruiter issues the "Maintain Companies" command.
2. Recruiter chooses whether they want to add a new company or modify the profile of an existing company.
3. To add a new client company, see section "Create Company."
4. To modify the profile of a company, see section "Modify Company Profile."

### Section—Create Company

1. Recruiter issues the "Create Client Company" command.
2. Recruiter enters the company profile including the company name, phone number, address, city, state/province, zip/postal code, and any other notes relevant to that company.
3. The Recruiter saves the information.
4. System Response—The company is added to the list of companies.

#### Alternative Courses

Line 3—Invalid data such as incomplete phone number or blank company name. Indicate error and allow reentry.

Section—Modify Company Profile

1. The Recruiter selects the company from the list of client companies and issues the "Modify Company" command.
2. System Response—Shows the company profile of the selected company.
3. Recruiter modifies the selected company's profile.
4. Recruiter saves the information.
5. System Response—The company profile is updated.

**Alternative Courses**

Line 4—Invalid data such as incomplete phone number or blank company name. Indicate error and allow reentry.

## *Use Case—Maintain Applicants*

Section—Main

Actors—Recruiters

Purpose—Allows the agency to keep a profile on applicants with the intention of placing them into current and future positions within the client companies.

Overview—The agency manages a list of all applicants and keeps track of whether they are currently available for placement or not.

New applicants can come into the agency to fill out applications or send applications to the agency via fax, e-mail, or regular mail. These applications are submitted into the system by a Recruiter. This creates an applicant profile that includes name, address, e-mail, phone, skill set, and other relevant notes. New applicants are considered available.

The agency hopes to maintain relationships with applicants over a long period of time. It serves the agency well to place the same applicant multiple times over the course of his or her career. The system must handle the changes to an applicant's profile such as a change in name, address, phone number, skills, etc.

The agency will maintain a history on all applicants. Therefore, they are not to be deleted from the system.

### Typical Course of Events

1. This use case begins when a Recruiter issues the "Maintain Applicants" command.
2. Recruiter chooses whether they want to submit an application that they received or make a change to an applicant profile.

3. To add a applicant, see section "Submit Application."

4. To modify the profile of an applicant, see section "Modify Applicant Profile."

## Section—Submit Application

1. Recruiter issues the "Fill Out Application" command.

2. System Response—Supplies an application form.

3. Recruiter enters the new applicant's first name, last name, address, city, state/province, zip/postal code, phone, e-mail, whether they are currently available (yes or no) and other relevant notes about the applicant.

4. Recruiter issues the "Select Skills" command.

5. System Response—Shows the list of industry skills maintained by the Agency (see the "View Skills" Use Case).

6. Recruiter selects the skills that apply to the applicant.

7. Recruiter saves the application.

8. System Response—Applicant is created from the information in the application form.

9. System Response—The applicant is inserted in the list of applicants.

**Alternative Courses**

Line 7—Recruiter is missing critical data. May cancel submitting the application.

Line 8—Application is invalid due to incorrect profile data (e.g., blank name, improper phone format, etc.) or no skills selected. Indicate errors and allow reentry.

## Section—Modify Applicant Profile

1. Recruiter selects the applicant from the applicant list and issues the "Modify Applicant Profile" command.

2. System Response—Shows the applicant's profile including the name (first, last), address, city, state/province, zip/postal code, phone, e-mail, whether they are currently available (yes or no), and other relevant notes.

3. Recruiter modifies the applicant's profile as appropriate.

4. Recruiter issues the "Select Skills" command.

5. System Response—Shows the list of industry skills maintained by the Agency (see the "View Skills" Use Case).

6. Recruiter adds and removes the skills of the applicant as appropriate.

7. Recruiter saves the application.

8. System Response—Applicant profile is updated.

9. System Response—The applicant is inserted in the list of applicants.

#### Alternative Courses

Line 4—Applicant may not need their skill set modified. Skip Lines 4, 5, and 6.

Line 7—Recruiter may be missing critical data. May cancel submitting application.

Line 8—Application is invalid due to incorrect profile data (e.g., blank name, improper phone format, etc.) or no skills selected. Indicate errors and allow reentry.

## *Use Case—Maintain Positions*

### Section—Main

Actors—Recruiters

Purpose—Allows the Agency to view, open, and cancel advertised positions.

Overview—The Agency keeps a list of positions advertised by companies. The agency is asked by a client company to supply applicants for a new position at the company. If the company is new to the agency, they are entered into the system (see "Maintain Companies" use case). Otherwise, a Recruiter enters the information about the position including the position's title, job description, company name, date the position opened, contact person, phone, and the required skills.

Once a position is closed, it is not reopened. Instead, a new position is created in the manner stated above.

Sometimes a company cancels a position because it was filled by someone supplied by another agency or by someone that the company found on their own. Other times the position is eliminated because a department is eliminated or a project is postponed or cancelled. Instead of deleting the position, the system will simply mark it closed, store the reason for closure, and the close date.

## Typical Course of Events

1. This use case begins when a Recruiter issues the "Maintain Positions" command.
2. Recruiter chooses whether they want to open a position, view positions, or cancel a position.

### Section—View Positions

## Typical Course of Events

1. A Recruiter issues the "View Positions" command.
2. System Response—Shows a list of client companies.
3. The Recruiter selects a company.
4. System Response—All positions for this company are displayed in a list.
5. The Recruiter selects the position from a position list.
6. System Response—The position's complete profile is displayed—Position Title, Description, Company Name, Contact Name, Contact Phone, Status, Date Position Opened, Date Position Closed, Reason Closed, Required Skills, and Notes.

### Section—Open Position

## Typical Course of Events

1. A Recruiter issues the "Open Position" command.
2. System Response—A list of client companies is shown.
3. Recruiter selects the company.
4. Recruiter enters information about the position.
5. Recruiter issues the "Select Skills" command to assign the required skills for the position.
6. System Response—Shows the list of industry skills.
7. Recruiter selects the required skills from the list.
8. Recruiter saves the position.
9. System Response—Position is given the status "Open" and is stored.

#### Alternative Courses

Line 8—Position is invalid (no skills selected, etc.);indicates error. May cancel opening the position.

## Section—Cancel Position

### Typical Course of Events

1. A Recruiter issues the "Cancel Position" command.
2. System Response—Show the list of open positions.
3. Recruiter selects the position to close from the list of open positions.
4. System Response—Show the position's complete profile. Ask for the reason the position is being closed and the close date.
5. Recruiter enters the reason why the position is closed and the date it was closed.
6. Recruiter saves the information.
7. System Response—Sets the position status to "Closed", sets the reason for closure and the close date to the ones provided by the recruiter, and stores the position.

### *Use Case—Find Positions for Applicants*

Actors—Recruiters

Purpose—To help Recruiters become more efficient in finding positions that best suit the applicants.

Overview—Recruiters work closely with applicants. They want to place them before someone else does or before they find employment on their own. To help in placing an applicant, the Recruiters need a list of open positions for which the applicant is qualified. The positions are sorted by best fit (i.e., the position with the highest percentage of matching skills on top).

### Typical Course of Events

1. This use case beginbs when a Recruiter issues the "Find Positions for Applicant" command.
2. System Response—Display the list of available applicants.
3. Recruiter selects an applicant from the list of available applicants and issues the "Find Possible Positions" command.
4. System Response—Searches the list of open positions and selects only those for which the applicant is qualified (i.e., positions where the

applicant has one or more skills among the required skills for the position).

5. System Response—Display the positions sorted by best fit.

### *Use Case—Maintain Skills*

Actors—Database Administrator

Purpose—To maintain a list of skills relevant to the industry of the agency's client companies. This list is determined by the staffing requirements of the client companies and is used to provide consistency when assigning skill sets to positions and applicants.

Overview—Typically, the agency staffs positions with titles such as programmers, analysts, system administrators, and database administrators. The Recruiters focus on finding applicants with skill sets that are relevant to these types of positions. For example, programmers could list Visual Basic, Java, and C++ in their skill set and DBAs could list SQL Server, Informix, and Oracle. Examples of skills added within the last 5 years are Java, MTS, and IIS.

The niche of the agency is high-tech, so the agency must continually adapt to fast changing skill sets required in positions posted by client companies. A skill needs to be added to the list of industry skills when a company posts a position requiring a skill that the agency has never dealt with in the past.

### Typical Course of Events

The Database Administrator handles this function directly using Microsoft Access.

### *Use Case—View Skills*

Actors—Recruiters

Purpose—To allow Recruiters to view skills for positions or applicants.

Overview—Typically, the agency staffs positions with titles such as programmers, analysts, system administrators, and database administrators. The Recruiters focus on finding applicants with skill sets that are relevant to these types of positions. For example, programmers could list Visual Basic, Java, and C++ in their skill set and DBAs could list SQL Server, Informix, and Oracle. Examples of skills added within the last 5 years are Java, MTS, and IIS.

### Typical Course of Events

1. This use case begins when a Recruiter issues the "View Skills" command.

2. System Response—Shows the list of industry skill titles.

## *Use Case—Apply for Position*

Actor—Applicant

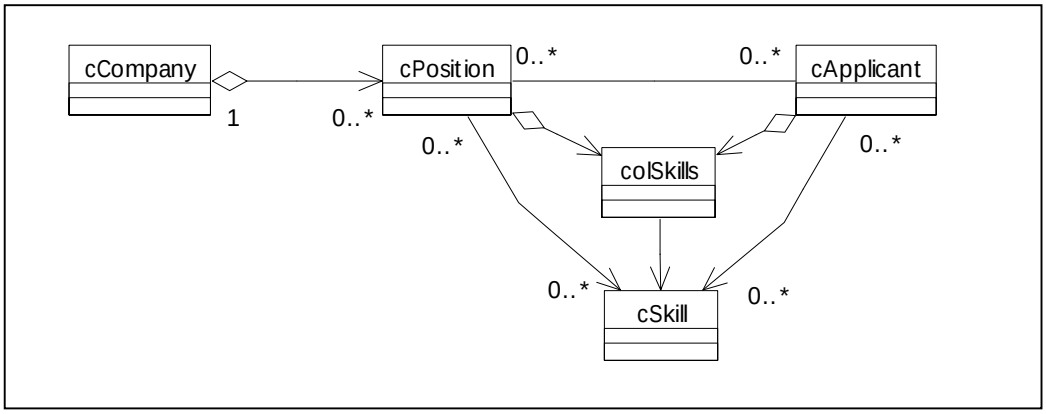
Purpose—To allow an applicant, via the Web, to apply for a position.

Overview—Virtual Placement Agency realizes that many of its registered applicants use the Web as a tool to search for jobs. Therefore, Virtual Placement Agency has "opened up" itself via the Web to its applicants. It allows the applicants to browse all the positions currently available and apply for only those they are interested in.

## Typical Course of Events

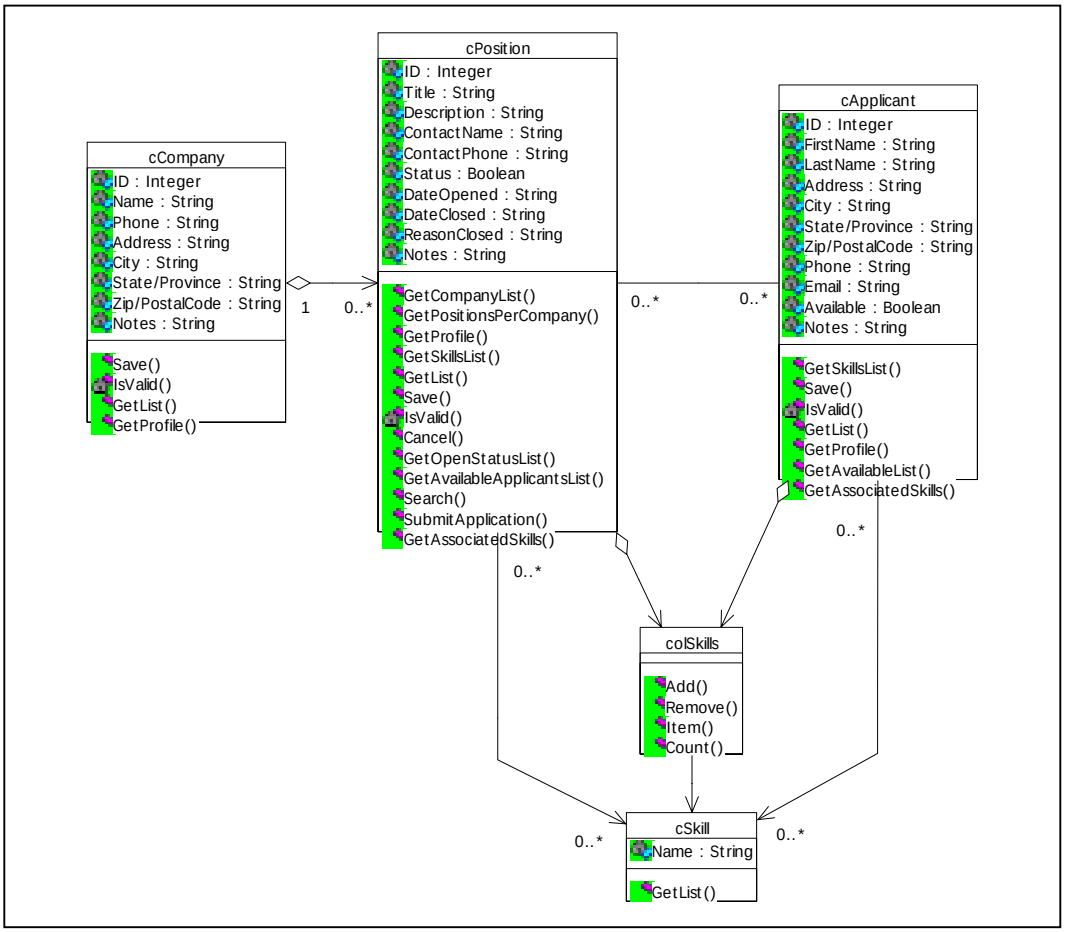
1. This use case begins when an applicant issues the "View List of Positions" command.
2. System Response—A list of all the agencies' positions is displayed.
3. The Applicant selects a position to display.
4. System Response—A position profile is displayed.
5. The Applicant issues the "Apply" command.
6. System Response—The current date and position ID are displayed.
7. The applicant enters their ID and submits the information.
8. System Response—A message is displayed indicating the information was saved.

# 3. Object Model




Virtual Placement Agency Object Model

# 4. Implementation Class Diagram




Virtual Placement Agency Implementation Class Diagram

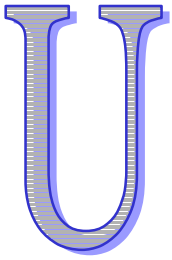
# 5. Table of Business Object Methods

| Class | Method | Description |
|---|---|---|
| cCompany | Save() | Called when the new or modified client company information needs to be updated to the database. |
| | IsValid() | Private method called by the Save() method; Validates the data being saved to avoid database data conflicts. |
| | GetList() | Called when a list of all client companies is required. |
| | GetProfile() | Used to retrieve a specific client company's profile. |
| cPosition | GetCompanyList() | Called when list of all the client companies is required. Calls the GetList from cCompany. |
| | GetPositionsPerCompany() | Used to retrieve all positions associated with a selected company. |
| | GetProfile() | Used to retrieve a specific position profile. |
| | GetSkillsList() | Called when list of all skills is required. Calls the GetList() method from cSkill. |
| | GetList() | Called when list of all positions is required. |
| | Save() | Called when the new or modified position information needs to be updated to the database, including associated skills. |
| | IsValid() | Private method called by the Save() method; Validates the data being saved to avoid database data conflicts. |
| | Cancel() | Called when cancel position command is issued. Updates position status (closed/open), saves the reason why position closed and date closed. |
| | GetOpenStatusList() | Called when list of positions with an open status is required. |
| | GetAvailableApplicantsList() | Called when list of available applicants is required. Calls the GetAvailableList() method from cApplicant. |
| | Search() | Searches the database for positions with a matching skills. Called when the find possible positions command is issued. |
| | SubmitApplication() | Called by the internet based application (Active Server Pages) when positions are applied for via the web; adds the current date, position ID and applicant ID and saves the data to the database. |
| | GetAssociatedSkills | Used to retrieve a list of associated skill from the database. A private method called by the GetProfile() method. |

| | | |
|---|---|---|
| cApplicant | GetSkillsList() | Called when list all of the skills is required. Calls the GetList() method from cSkill. |
| | Save() | Called when the new or modified applicant information needs to be updated to the database, including associate skills. |
| | IsValid() | Private method called on Save() method; Validates data being saved to avoid database data conflicts. |
| | GetList() | Called when list of all applicants is required. |
| | GetProfile() | Used to retrieve a specific applicant profile. |
| | GetAvailableList() | Called when list of available applicants is required. |
| | GetAssociatedSkills() | Used to retrieve a list of associated skill from the database. A private method called by the GetProfile() method is called. |
| cSkill | GetList() | Called when list of all skills is required. |
| colSkills | Add() | Adds object references to skills collection. |
| | Remove() | Removes object references to skills collection. |
| | Item() | Returns a specific item in the collection. |
| | Count() | Returns the total number of items in the collection. |

# 6 Table of Object Relationships

| Class | Relationship | Description |
|---|---|---|
| cCompany – cPosition | Aggregation | The Company "may have " 0 or many positions. Without the company, there would not be a positions. |
| cPosition – cApplicant | Association | Each applicant can apply for many positions; each position can have more than one applicant. |
| cApplicant – cSkill | Unidirectional Association | An applicants "may have" 0 or many skills. |
| cPosition – cSkill | Unidirectional Association | An position "may have" 0 or many associated skills. |



# 1. Application Goals and Objectives

## *Main Purpose*

The goal of this project is to develop an enterprise application that will allow the recruiting staff at Virtual Placement Agency (VPA) to manipulate information on job openings and placements managed by the agency. This will require the tracking of job postings, hiring companies, and applicants.

## *Business Objectives*

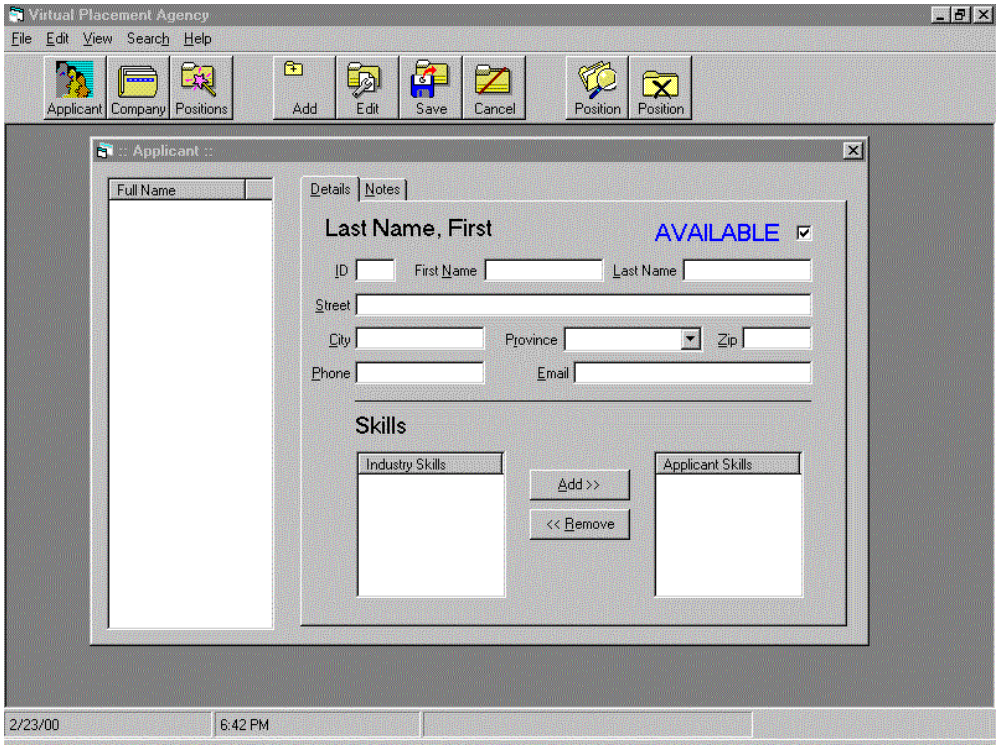
- To track and maintain the three major components of the business : Companies, Applicants, Positions.
- Match applicants to available positions.
- Allow applicants looking for employment to access information on available job openings by accessing the VPA's website.

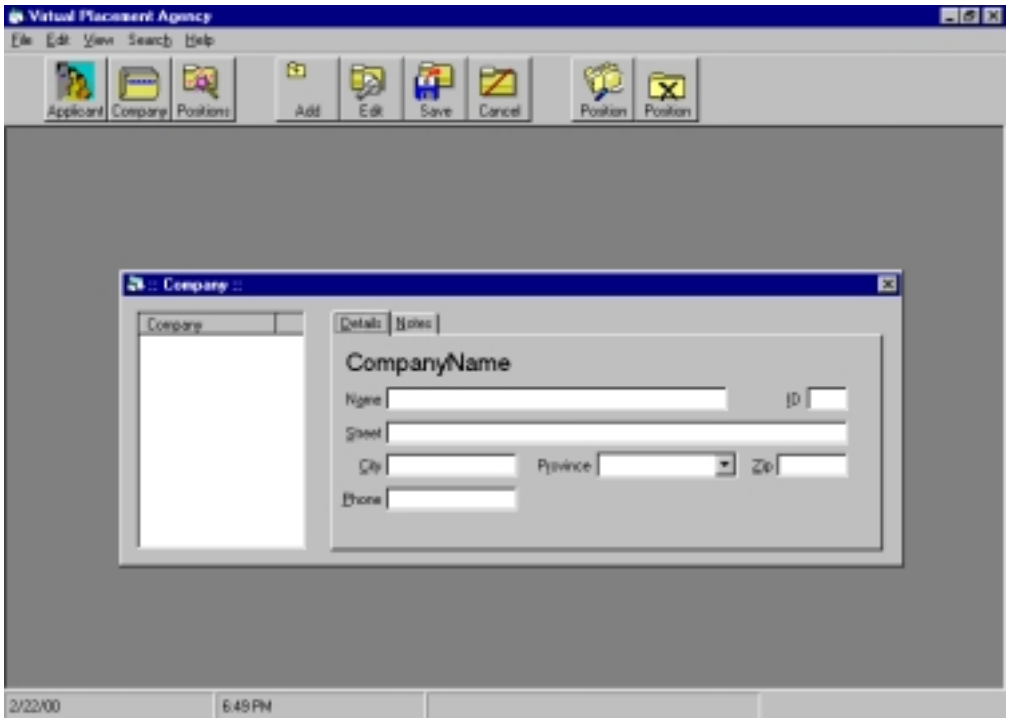
## *Technical Objectives*

- Formalize the design of the software using UML modeling techniques and the GUIDS methodology.
- Follow and object-oriented design methodology to ensure the application is properly defined before it is built.
- Use Microsoft Windows GUI Standards to develop a user-friendly interface.
- Provide a software environment that operates effectively with Microsoft Windows NT Server, Windows 98 and Access 97.
- Design scalable application that can be expanded as the agency grows.
- Develop a distributed component-based system that runs on a client/server network .
- Develop Active X components, using Visual Basic 6, that are managed by using Microsoft Transaction Server (MTS).
- Creation of the web enabled application using Active Server Pages (ASP).

# 2. User Interface: Maintain Companies

## *Screen Shot*



## *Table of Components*

| Control | Name of Control | Description |
|---|---|---|
| Menu | File→Close All Forms | Close all active forms |
| | File→Save | Save all fields of the active form to the database |
| | File→Exit | Unload all forms and exit |
| | Edit→Close Positions | Opens the Close Position form. |
| | View→Company | Opens the company form |
| | View→Applicant | Opens the applicant form |
| | View→Position | Opens the position form |
| | View→Industry Skills | Opens the industry skills form. |
| | Search→Positions Form Applicants | Open the search position form |
| | Help→About | Shows information about version number. |
| Tool Bar | Add | Clears and enables all fields of the active form and allow the recruiter to enter new information |
| | Save | Saves all fields of the active form to the database |
| | Edit | Enables all fields of the active form and allows recruiter to edit information |
| | Cancel | Cancels the current entry. |
| | Find Positions | Searches open positions for which the applicant is qualified |
| | Close Position | Opens the close position form |
| List View | Company | Displays a list of all client companies |
| Tab Strip | Company | Allows user to flip to and from the notes form view. |
| Combo Box | Province | Used to select and display the province or state. Note: The combo box contains a complete list of Canadian provinces and states. This field will be filled by selecting a province /state from the list. |
| Text Box | ID | This number is assigned automatically when a new company is added. This will ensure unique identifiers are assigned to each company. |
| | Name | Displays the name of the selected company; allows recruiter to enter information to this field. |
| | Street | Displays the company's street address; allows a recruiter to enter information in this field. |
| | City | Display the city in which the company is located; allows a recruiter to enter information in this field. |
| | Zip | This field will display the postal code. If a nonnumeric code is added, it will be unformatted before the text is stored to the database. When the data is retrieved from the database it will be reformatted. (e.g. A#A-#A#) |

| | | |
|---|---|---|
| | Phone | This field will automatically format numbers to the required layout. The format mask will also stop the entry of any nonnumeric key. (e.g. (###) ###-####) |
| | Notes | Multiple lines of text can be added. Up to 32 kilobytes of text can be added |

## *Feature List*

### View Companies

To view the company form the user may select the company button from the tool bar, select View → Company from the menu bar or use the shortcut key Ctrl-W.

On the loading of the form the first record set in the database will be loaded, populating the data fields. The list view will also be populated with a complete list of all the companies in the database.

Selecting and viewing company profiles:
To view a company's profile, use the mouse to select the company name in the list view. Once the item has been highlighted the company profile is returned from the database.

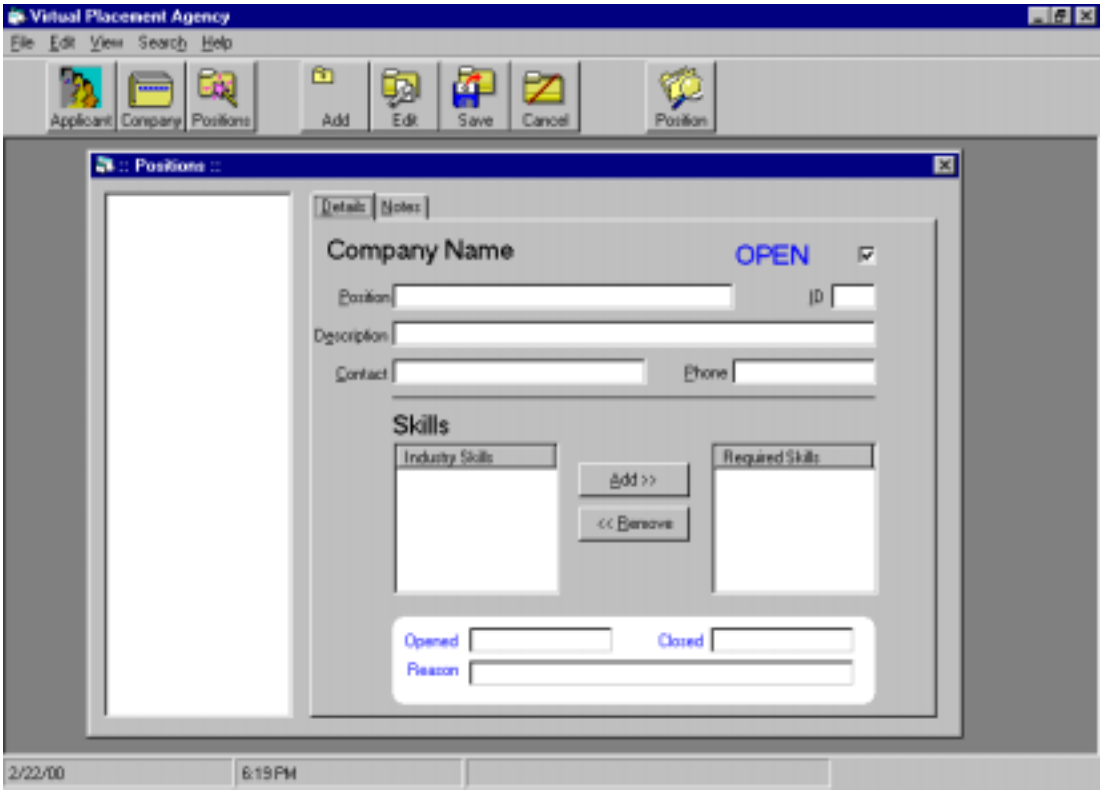
### Adding, Editing and Saving Company Profiles

To add a new company to the system select the add button from the tool bar. The system will respond by clearing the form and enabling the text boxes for data entry. The ID field will remain disabled as this value is supplied by the database.

When all the information has been entered the data needs to be validated and saved to the database. This is accomplished by selecting the save button on the tool bar, by selecting File →Save from the menu bar or by using the shortcut key Ctrl-S. If there are any required fields that have not been entered correctly the system will return to the form and the labels of the problem fields will be in red. Once the invalid fields have been corrected, select save again to save the data to the database. On a successful save, the new company will be added to the company list view and the text boxes will be disabled.

If a selected record set requires editing, the edit function is activated. This can be accomplished by selecting the Edit button from the toolbar. The system responds by enabling the text boxes. On the completion of the edit, the save function is used to save the data to the database. Before saving the data the system will validate all the data fields.

## 3. User Interface: Maintain Positions




### *Table of Components*

| Control | Name of Control | Description |
|---|---|---|
| Tree View | Company | Displays a list of client companies. |
| Tab Strip | Position | Allows user to flip to and from the notes form view. |
| Text Box | ID | Entry of the company ID.<br>Note: The AutoNumber of the database fills this field. This will ensure unique identifiers are assigned to each company. |
| | Position | Display the selected position title; allows the recruiter to enter information to this field. |
| | Description | Display the selected position's job description; allow recruiter to enter information to this field. |
| | Opened | Display the selected position's Open Date; allow recruiter to enter information to this field. e.g. MM/DD/YY |
| | Closed | Display the selected position's Close Date; allow recruiter to enter information to this field. e.g. MM/DD/YY |
| | Reason | Displays the reason a position has been closed. |

| | | |
|---|---|---|
| | Contact | Displays the name of a contact person at the company that is offering the position; allows the recruiter to enter information to this field |
| | Phone | This field will automatically format numbers to the required layout. The format mask will also stop the entry of any key other than numbers and backspace.<br>e.g. (###) ###-#### |
| | Notes | On selection, multiple lines of text can be added. Up to 32 kilobytes of text can be added |
| Label | Company | Displays the name of a selected company. |
| Check Box | Status | Displays the position status; allows the recruiter to select the status of a position. If unchecked, the word closed appears. |
| List View | Skill | Displays a list of all the industry skills. |
| | Required Skills | Displays a list of required skills selected from the industry skills list box. |
| Command Button | Add | Adds selected industry skill to list of required skills |
| | Remove | Removes selected industry skill from list of required skills |
| | Close Position | Removes position from list of open positions and sets position status to Closed |

## *Feature List*

### View Positions

To view the positions form the recruiter selects Positions from the toolbar, View→Position from the menu bar or uses the shortcut keys Ctrl+E. A list of client companies is made available to the recruiter. Once a company is selected, all positions for this company are displayed. A specific position may be selected from the list in order to display that position's profile.

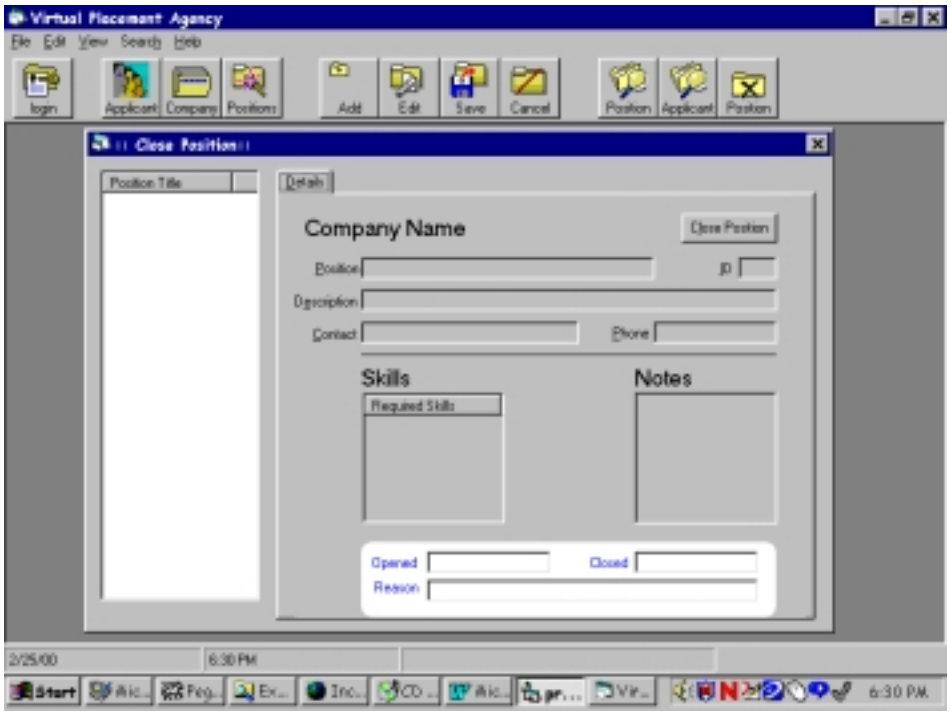
### Open Positions

The recruiter selects the positions from and then a client company. Once a company is selected, the recruiter selects the Add button from the toolbar to enters information on the new position. The list of industry skills allows the recruiter to choose the associated skills. The recruiter may then save the information to the database and the new position is added to the list of positions in the company. The position is automatically given an "Open" status. The cancel button on the toolbar is used to cancel the opening of a new position.

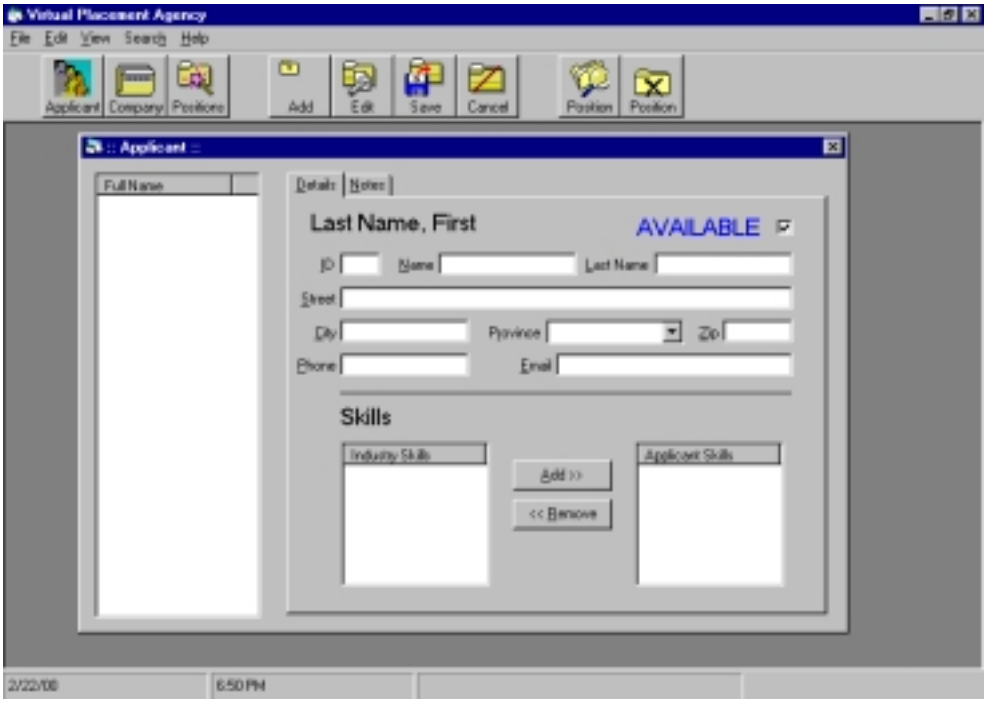
### Cancel Position

Upon selecting close position from the menu bar (Edit→Close Positions) or selecting Close Position on the toolbar, the Close Positions form is loaded and a list of open positions is displayed. The recruiter selects the position that is to be closed, displaying the profile of the position. The recruiter presses the Close Position button on the form. and the system prompts for the reason for closing the position as well as the date of

closure. This information may be is saved, setting the status of the the position to Closed and removing the position from the list of open positions.




## 4. User Interface: Maintain Applicants



## *Table of Components*

| Control | Name of Control | Description |
|---|---|---|
| Tab Strip | Applicant | Allows user to flip to and from the notes form view. |
| Text Box | ID | Displays the selected applicant's ID; allows the recruiter to enter information into this field. |
| | First Name | Displays the selected applicant's First Name; allows the recruiter to enter information into this field. |
| | Last Name | Displays the selected applicant's Last Name; allows the recruiter to enter information into this field. |
| | Street | Displays the selected applicant's Street Address; allows the recruiter to enter information into this field. |
| | City | Displays the selected applicant's City of residence; allows the recruiter to enter information into this field. |
| | Zip | This field will display the postal code or zip code. If a nonnumeric code is added of 7 characters, it will be unformatted before the text is stored in the database. On return of the data from the database it will be reformatted. e.g. A#A-#A# |
| | Phone | This field will automatically format numbers to the required layout. The format mask will also stop the entry of any key other than numbers and backspace. e.g. (###) ###-#### |
| | Email | Displays the selected applicant's Email Address; allows the recruiter to enter information into this field. |
| | Notes | Displays notes or memo of selected applicant; allows the recruiter to enter information into this field. Up to 32 kilobytes of text can be added. |
| Check Box | Available | Displays the selected applicant's availability. |
| Combo Box | Province | Entry and display of the province/state in which the company is located. Note: The combo box contains a complete list of Canadian provinces and US states. This field will be filled by selecting a province /state from the list. |
| List View | Applicant | Lists all applicants in the agency database. By selecting an applicant in the list the corresponding profile is displayed. Note: This field is populated by the database. |
| | Applicant Skill | Displays skills associated with applicant. |
| | Skill | Displays all available industry skills. |
| Command Button | Add | Adds selected industry skill to list of applicant skills. |
| | Remove | Removes selected industry skill from list of applicant skills. |

## *Feature List*

### View Applicants

To view an applicant form the user may select the applicant button on tool bar, select View→ Applicant from the menu bar or use the shortcut keys Ctrl+Q.

On loading the form the first record in the database will be loaded, populating the data fields. The applicant list will also be populated with a complete list of all applicants.

To view a specific applicants profile, select their name from the applicant list. Once the name has been highlighted in the list the applicant profile is returned from the database.

### Add Applicants

Pressing Add button on the tool bar, clears the text boxes and enables data entry , allowing the user to add a new applicant.

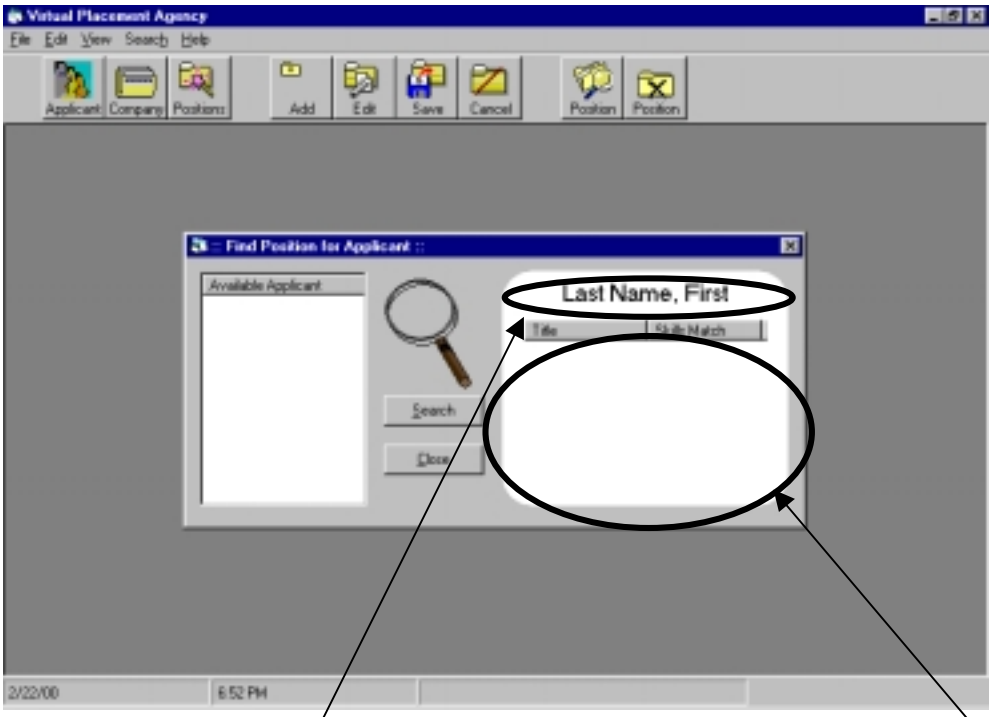
To add information on a new applicant's skills, the industrial skill is selected from the list box. The Add button on the form is then used. to associated the selected skill with the applicant. A list of associated skills is displayed in the applicant skill list. If an incorrect skill is added it can be removed by selecting the skill in the applicant skill list and pressing the remove button.

Saving this new information will add the applicant information to the database and add the applicant to the list of applicants. Selecting cancel from the toolbar will return the user to the first applicant profile in the database without saving the new applicant profile.

### Edit Applicant

Pressing the Edit button on the tool bar will enable data entry into the applicant's profile. Selecting save will then save the changes to the database. During the edit proccess cancel can be pressed to exit the applicant's profile without saving the changes.

# 5. User Interface: Find Positions for Applicant




Applicant's name is displayed here, and the positions with matching skills are displayed here.

## *Table of Components*

| Control | Name of Control | Description |
|---|---|---|
| List View | Applicants | Displays a list of all applicants. |
| | Positions | Displays a list of positions and the number of matching skills. |
| Command Button | Search | Searches for positions with some of the same skills as the selected applicant. |
| | Close | Takes the user back to the position form and the profile will be displayed |

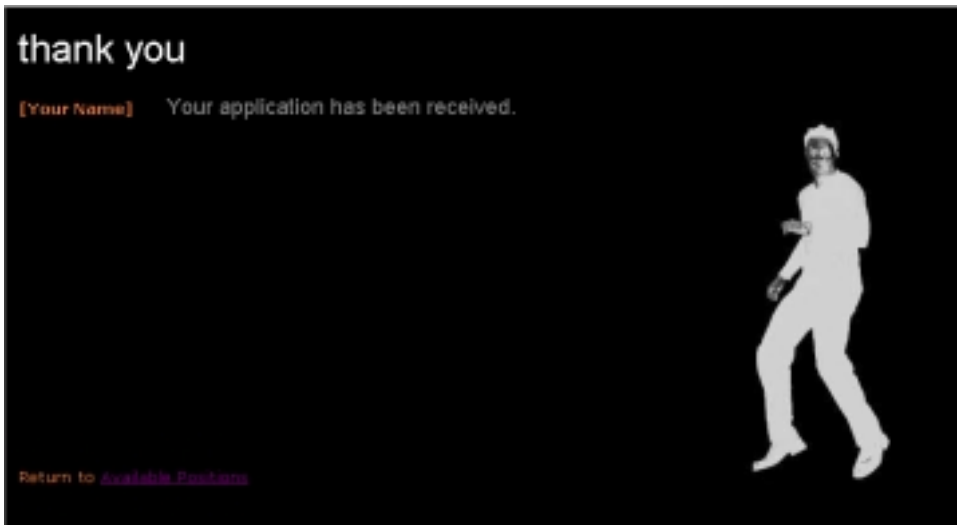
## *Feature List*

On form load a list of available applicants will populate the first list box. The user may select an applicant by selecting a name from the list. Pressing the Search button will populate the second list box with a list of open positions for which the selected applicant has a matching skill set. The positions are listed with the highest percentage of matching skills on top.

The recruiter may find details of one of the matching positions by highlighting the position and clicking the Close button. This will load the Positions form, displaying all the details associated with the selected position.

## 6. User Interface: Apply for Online Position





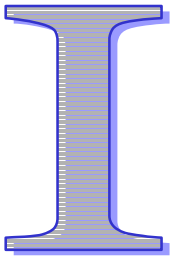

## *Table of Components*

| Control | Name of Control | Description |
|---|---|---|
| Screen 2: List Box | Position | Display a list of all the agencies' open positions. |
| | Position Skills | Displays a list of associated skills for selected position. |
| Screen 2: Text Box | Position ID | Displays the selected position ID |
| | Position Title | Displays the selected position title |
| | Description | Displays the selected position description |
| | Contact Name | Displays the selected position contact name |
| | Contact Phone | Displays the selected company phone number |
| | Date Opened | Displays the selected position's date opened. |
| Screen 2: Button | Apply | On selecting the apply button the selected positions ID and title fields is passed to the application page and the application page loaded. |
| Screen 3: Text Box | Position ID | Displays the selected position ID. |
| | Position Title | Displays the selected position title. |
| | Applicant ID | Allows the applicant to add their ID. |
| Screen 3: Button | Apply | Sends the information to the database. |

## *Feature List*

Applicant enters the virtual placement agency home page and clicks the 'Enter' hyperlinked text. The second screen is loaded and a list of all VPA positions is displayed in the list box. The applicant then selects a position of interest. The selected position profile is then displayed in the text fields. The applicant can then issues the apply for the position by clicking the 'Apply' button'.

The third screen is then loadeddisplaying the current date and position ID. To apply for this position, the applicant enters their ID and then clicks the 'apply' button. A thank you screen is displayed with applicant's full name displayed after application is received.



# 1. Application Goals and Objectives

## *Main Purpose*

The goal of this project is to develop an enterprise application that will allow the recruiting staff at Virtual Placement Agency (VPA) to manipulate information on job openings and placements managed by the agency. This will require the tracking of job postings, hiring companies, and applicants.

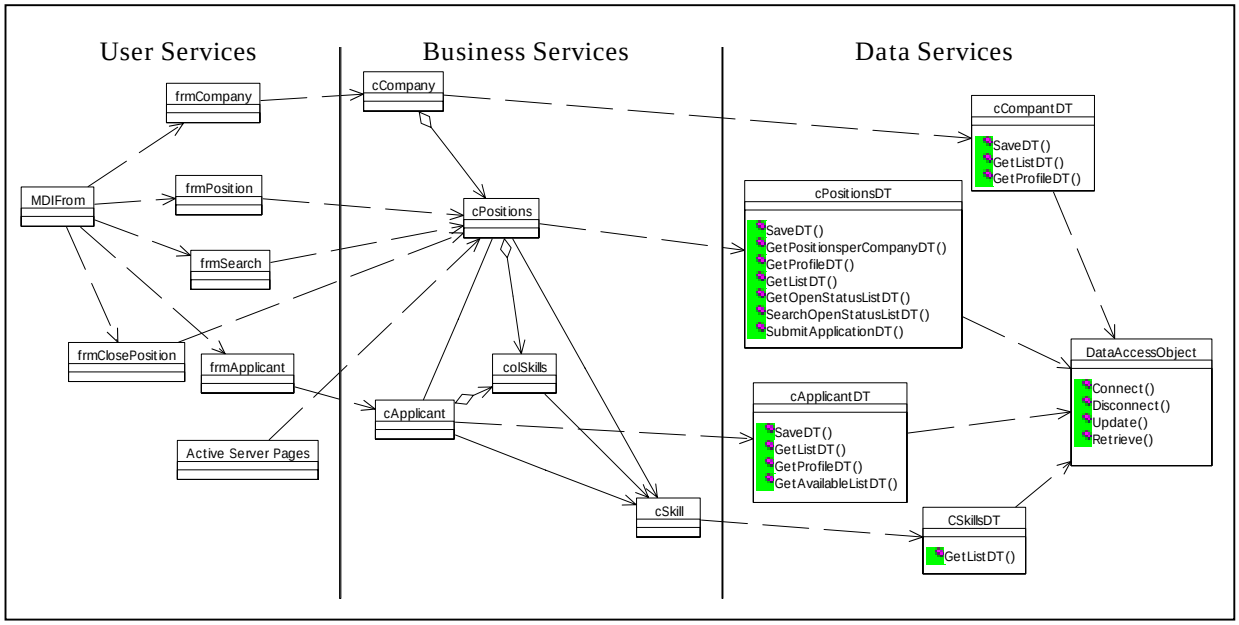
## *Business Objectives*

- To track and maintain the three major components of the business : Companies, Applicants, Positions.
- Match applicants to available positions.
- Allow applicants looking for employment to access information on available job openings by accessing the VPA's website.

## *Technical Objectives*

- Formalize the design of the software using UML modeling techniques and the GUIDS methodology.
- Follow and object-oriented design methodology to ensure the application is properly defined before it is built.
- Use Microsoft Windows GUI Standards to develop a user-friendly interface.
- Provide a software environment that operates effectively with Microsoft Windows NT Server, Windows 98 and Access 97.
- Design scalable application that can be expanded as the agency grows.
- Develop a distributed component-based system that runs on a client/server network .
- Develop Active X components, using Visual Basic 6, that are managed by using Microsoft Transaction Server (MTS).
- Creation of the web enabled application using Active Server Pages (ASP).

# 2. Application Architecture




Three Tiered Service Model

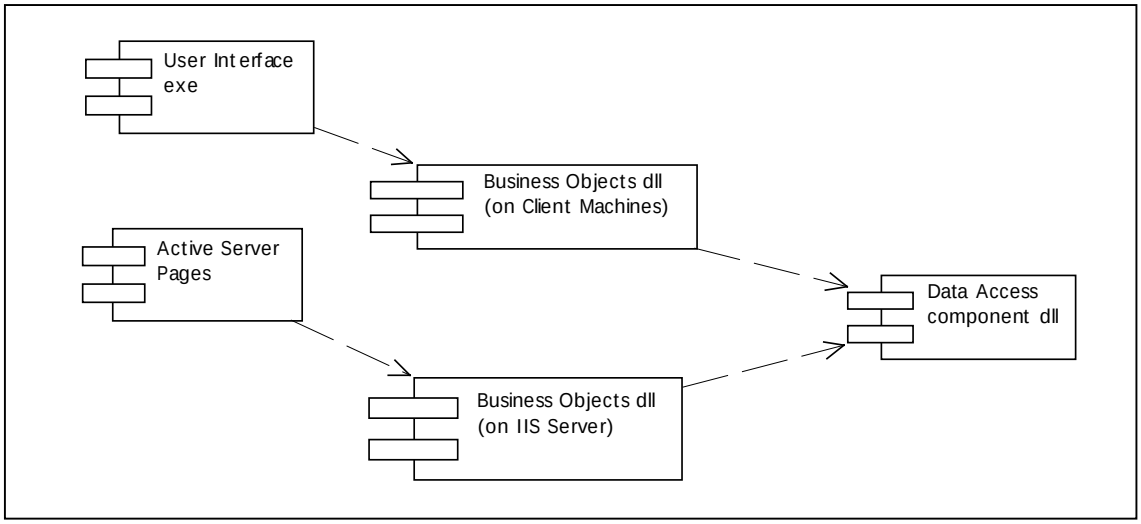
The classes in the Data Services section are stateless (ie they have no properties). The difference between the methods in the Data Services classes and those of the Business Services is that the methods in Data Services classes are SQL statements, which retrieve data from or update data to the database via the DataAccess Object. The DataAccess object is responsible for the actual data transfer with the database. The SQL statements are discussed in the D document.

| Class | Method | Description |
|---|---|---|
| DataAccess Object | Connect() | Opens a connection to the database |
| | Disconnect() | Closes the database connection |
| | Retrieve() | Uses the open database connection to retrieve data from the database |
| | Update | Uses the open database connection to send data into the database. |

# 3. Component Distribution

The proposed system will compose of 5 components as shown in the component diagram. The components will be distributed as follows. The user interface exe and the business object dll will both reside on the client machines. The Data Access Component dll will be located on the NT Server with the database. To allow web access Active Server Pages (ASP) are being used to pass information over the internet. These ASP's need to connect to the business objects before being passed to the data access component.

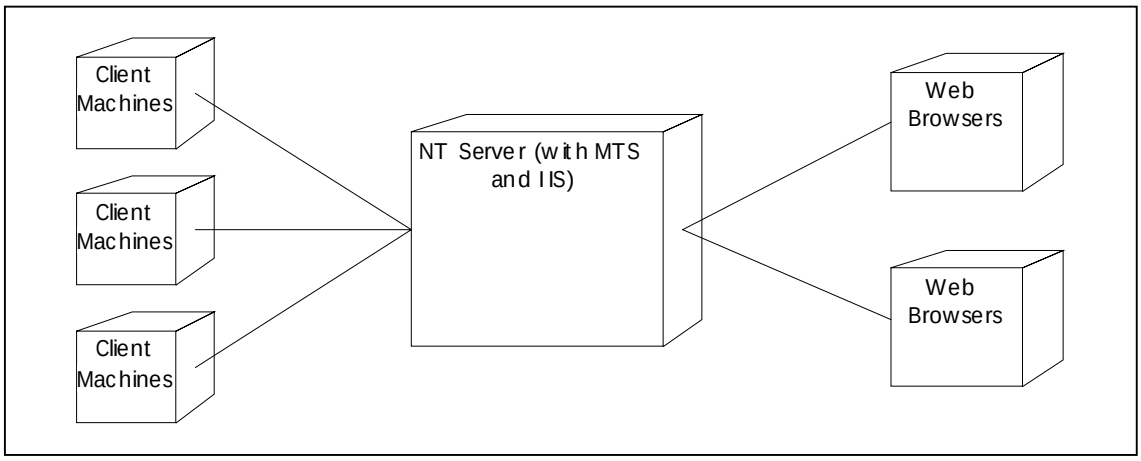
Therefore a second business object dll component will be located on the internet information server (IIS) to service the ASP's.




## *Component Diagram Elements*

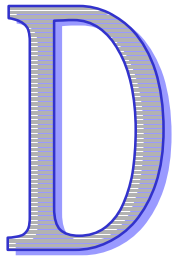
| Component Name | Objects Realized | Description |
|---|---|---|
| User Interface exe | MDIform, frmPosition, frmCompany, frmSplash, frmSearch, frmApplicant, frmSkills | Applications used by user - connected to business object dll (on client machine). |
| Business Object dll (on Client Machines) | cPositions, cCompany, cApplicant, cSkills | Sends and receives information from the User Interface exe. Passes information to Data Access Object dll. |
| Active Server Page | Active Server Pages | Displayed in internet browsers as an interactive web page. |
| Business Objects dll (on IIS Server) | cPositions | Sends and Receives information from active server page. Passes information to Data Access Object dll. |
| Data Access Component dll | cPositionsDT, cCompanyDT, cApplicantDT, cSkillsDT, DataAccess Object | Sends and receives data from both business objects dll. |

# 4. Deployment Diagram




## *Deployment Diagram Elements*

| Node | Description |
|---|---|
| Client Machines | Contains the user interface and business object components. |
| NT Server (with MTS and IIS) | Contains the database and data access objects managed by Microsoft Transaction Server (MTS). The server will also contain the Active Server Pages (ASP) and a business object component to service the ASP's. |



# 1. Application Goals and Objectives

## *Main Purpose*

The goal of this project is to develop an enterprise application that will allow the recruiting staff at Virtual Placement Agency (VPA) to manipulate information on job openings and placements managed by the agency. This will require the tracking of job postings, hiring companies, and applicants.

## *Business Objectives*

- To track and maintain the three major components of the business : Companies, Applicants, Positions.
- Match applicants to available positions.
- Allow applicants looking for employment to access information on available job openings by accessing the VPA's website.

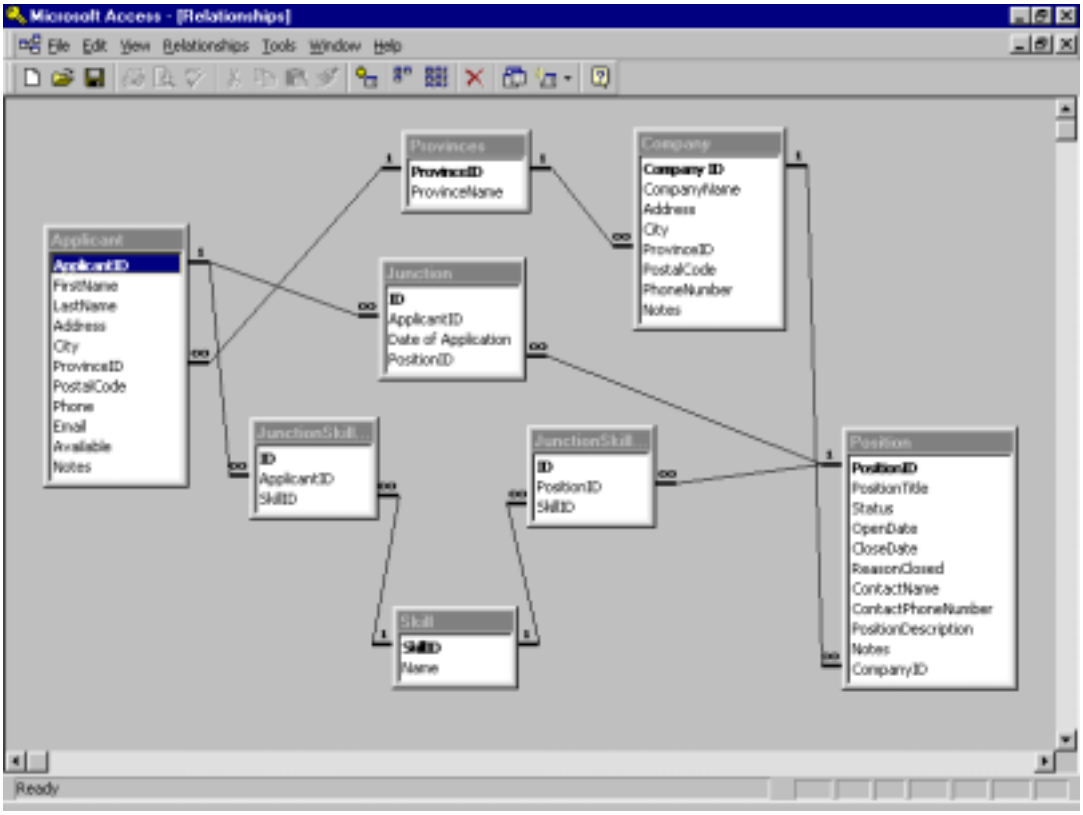
## *Technical Objectives*

- Formalize the design of the software using UML modeling techniques and the GUIDS methodology.
- Follow and object-oriented design methodology to ensure the application is properly defined before it is built.
- Use Microsoft Windows GUI Standards to develop a user-friendly interface.
- Provide a software environment that operates effectively with Microsoft Windows NT Server, Windows 98 and Access 97.
- Design scalable application that can be expanded as the agency grows.
- Develop a distributed component-based system that runs on a client/server network .
- Develop Active X components, using Visual Basic 6, that are managed by using Microsoft Transaction Server (MTS).
- Creation of the web enabled application using Active Server Pages (ASP).

# 2. Type of database

Due to the low user traffic and the scalability of the n-tiered design Microsoft Access will be used for the database. The alternative databases would significantly increase the application cost.

## 3. Entity Relationship Diagram




## 4. Table Fields and Descriptions

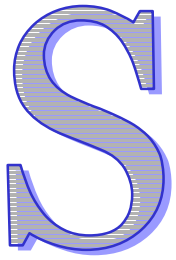
| Table | Field Name | Data Type | Key | Required |
|---|---|---|---|---|
| Applicant | Applicant ID | AutoNumber | Primary Key | Yes |
| | FirstName | Text | | Yes |
| | LastName | Text | | Yes |
| | Address | Text | | Yes |
| | City | Text | | Yes |
| | ProvinceID | Number | Foreign Key | Yes |
| | PostalCode | Text | | Yes |
| | Phone | Text | | Yes |
| | Email | Text | | Yes |
| | Available | Yes/No | | Yes |
| | Notes | Memo | | Yes |
| Company | CompanyID | AutoNumber | Primary Key | Yes |
| | CompanyName | Text | | Yes |
| | Address | Text | | Yes |
| | City | Text | | Yes |
| | ProvinceID | Number | Foreign Key | Yes |

| | | | | |
|---|---|---|---|---|
| | PostalCode | Text | | Yes |
| | PhoneNumber | Text | | Yes |
| | Notes | Memo | | Yes |
| Position | PositionID | AutoNumber | Primary Key | Yes |
| | PositionTitle | Text | | Yes |
| | Status | Text | | Yes |
| | OpenDate | Text | | Yes |
| | CloseDate | Text | | Yes |
| | ReasonClosed | Text | | Yes |
| | ContactName | Text | | Yes |
| | ContactPhoneNumber | Text | | Yes |
| | PositionDescription | Text | | Yes |
| | Notes | Memo | | Yes |
| | CompanyID | Number | Foreign Key | Yes |
| Skill | SkillID | AutoNumber | Primary Key | Yes |
| | Name | Text | | Yes |
| Province | ProvinceId | AutoNumber | Primary Key | Yes |
| | ProvinceName | Text | | Yes |
| Junction | ID | AutoNumber | Primary Key | |
| | ApplicantId | Number | Foreign Key | |
| | Date of Application | Date/Time | | |
| | PositionId | Number | Foreign Key | |
| Junction SkillApplicant | ID | AutoNumber | Primary Key | |
| | ApplicantId | Number | Foreign Key | |
| | SkillId | Number | ForeignKey | |
| Junction SkillPosition | ID | AutoNumber | Primary Key | |
| | PositionId | Number | Foreign Key | |
| | SkillId | Number | ForeignKey | |

## 5. List of Queries Required in the System

| Class | Method | Description |
|---|---|---|
| cCompanyDT | SaveDT() | Generates an SQL statement that will pass a company ID, as an argument and a profile to the data access object as a variant array. Depending on the ID the data access object will either update an existing record set or add a new record set. |
| | GetListDT() | Generates an SQL statement that will create and return a list of companies as a string array. |
| | GetProfileDT() | Generates an SQL statement that will pass a company ID as an argument and return specific company profile as a variant array. |
| cPositionDT | SaveDT() | Generates an SQL statement that will pass a position ID, as an argument and a profile to the data access object as a variant array. Depending on the ID the data access object will either update an existing record set or add a new record set. |

| | | |
|---|---|---|
| | GetPositionsper-CompanyDT() | Generates an SQL statement that will pass the selected company's ID as an argument and return specific list of company positions that are open. The data is returned as a string array. |
| | GetProfileDT() | Generates an SQL statement that will pass a position ID as an argument and return specific position profile as a variant array. |
| | GetListDT | Generates an SQL statement that will create and return a list of positions as a string array. |
| | GetOpen-StatusListDT() | Generates an SQL statement that will return a list of open positions as a string array. |
| | SearchOpenStatus-ListDT() | Generates an SQL statement that will search the open position list for a match to an applicant. The SQL statement will pass a list of skills as a string array and return a list of positions as a string array. |
| | Submit-ApplicationDT() | Generates an SQL statement that will pass the web based job application information to the database. The SQL statement will then match the user ID to the user ID's in the database. If a match is found the information is saved and the user name returned, if no match is found an error message is returned. The data will be passed as a variant array and the returned data passed as a string.. |
| cApplicantDT | SaveDT() | Generates an SQL statement that will pass an applicant ID, as an argument and a profile to the data access object as a variant array. Depending on the ID the data access object will either update an existing record set or add a new record set. |
| | GetListDT() | Generates an SQL statement that will create and return a list of applicants as a string array. |
| | GetProfileDT | Generates an SQL statement that will pass an applicant ID as an argument and return specific company profile as a variant array. |
| | GetAvailable-ListDT() | Generates an SQL statement that will return a list of available applicants. |
| DataAccess-Object | Connect() | Makes the connection to the database. |
| | Disconnect() | Disconnects from the database. |
| | Update() | Presents the SQL statement and data to the database. |
| | Retrieve() | Returns data requested by the SQL statement. |
| cSkillDT | GetListDT() | Generates an SQL statement that will create and return a list of skills as a string array. |



# 1. Application Goals and Objectives

## *Main Purpose*

The goal of this project is to develop an enterprise application that will allow the recruiting staff at Virtual Placement Agency (VPA) to manipulate information on job openings and placements managed by the agency. This will require the tracking of job postings, hiring companies, and applicants.

## *Business Objectives*

- To track and maintain the three major components of the business : Companies, Applicants, Positions.
- Match applicants to available positions.
- Allow applicants looking for employment to access information on available job openings by accessing the VPA's website.

## *Technical Objectives*

- Formalize the design of the software using UML modeling techniques and the GUIDS methodology.
- Follow and object-oriented design methodology to ensure the application is properly defined before it is built.
- Use Microsoft Windows GUI Standards to develop a user-friendly interface.
- Provide a software environment that operates effectively with Microsoft Windows NT Server, Windows 98 and Access 97.
- Design scalable application that can be expanded as the agency grows.
- Develop a distributed component-based system that runs on a client/server network .
- Develop Active X components, using Visual Basic 6, that are managed by using Microsoft Transaction Server (MTS).
- Creation of the web enabled application using Active Server Pages (ASP).

# 2. Code Estimates

| **Coding Estimates** | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | **Complexity** | | | |
| **Type** | **Component** | **High** | **Medium** | **Low** | |
| | | | | | |
| ***Business Servers - Data Centric*** | | | | | |
| | cCompanyDT | | 1 | | |
| | cPositionsDT | | 1 | | |
| | cApplicantDT | | 1 | | |
| | cSkills | | 1 | | |
| | Total Components | 0 | 4 | 0 | |
| | | | | | |
| | *Estimating Guideline (Hrs)* | *20* | *10* | *10* | |
| | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| | Total Business Servers - Data Centric | 0 | 40 | 0 | |
| | | | | | |
| ***Business Servers - UI Centric*** | | | | | |
| | cCompany | | 1 | | |
| | cPositions | | 1 | 1 | |
| | cApplicants | | 1 | | |
| | cSkills | | 1 | 1 | |
| | | | | | |
| | Total Components | 0 | 4 | 2 | |
| | | | | | |
| | *Estimating Guideline (Hrs)* | *15* | *10* | *5* | |
| | | | | | |
| | Total Business Servers - UI Centric | 0 | 40 | 10 | |
| | | | | | |
| ***Utility Servers*** | | | | | |
| | CDataAccess | 1 | | | |
| | CError | | 1 | | |
| | CUI | | 1 | | |
| | | | | | |
| | Total Components | 1 | 2 | 0 | |
| | | | | | |
| | *Estimating Guideline (Hrs)* | *30* | *15* | *8* | |
| | | | | | |
| | Total Utility Servers | 30 | 30 | 0 | |
| | | | | | |
| ***UI - Forms*** | | | | | |
| | frmCompany | | 1 | | |
| | frmPositions | | 1 | | |
| | frmApplicants | | 1 | | |
| | frmPositionClosed | | | 1 | |
| | frmMDIForm | 1 | | | |
| | frmSearch | 1 | | | |
| | Active Server Page (ASP) | 1 | | | |
| | | | | | |
| | | | | | |
| | Total Components | 3 | 3 | 1 | |
| | | | | | |
| | *Estimating Guideline (Hrs)* | *30* | *20* | *10* | |
| | | | | | |
| | Total UI - Forms | 90 | 80 | 10 | |
| | | | | | |
| ***UI - Web*** | | | | | |
| | Applicant.ASP | | 1 | | |
| | Applicantdb.ASP | | 1 | | |
| | | | | | |
| | | | | | |
| | Total Components | 0 | 2 | 0 | |
| | | | | | |
| | *Estimating Guideline (Hrs)* | *30* | *20* | *10* | |
| | | | | | |
| | Total UI - Web | **0** | **40** | **0** | |
| | | | | | |
| ***Total Hours*** | | ***120*** | ***230*** | ***20*** | ***370*** |

# 3. Implementation Plan

| | | Estimated Hours | | | |
|---|---|---|---|---|---|
| | | Principal | Software | Start | End |
| | Resource | Consultant | Developer | Week | Week |
| ***Software Development*** | | | | | |
| | | | | | |
| *Project Management* | | | | | |
| Project Tracking and Status Reporting | DDS | 40 | | 5 | 13 |
| Deliverable Review | DDS | 40 | | 5 | 13 |
| | | | | | |
| *Test Environment* | | | | | |
| Install hardware/software (e.g. IIS,MTS,SQL Sever, Oracle) | EGG | | | 5 | 5 |
| Configure software | EGG | | | 5 | 5 |
| Implement backup/restores | EGG | | | 5 | 5 |
| | | | | | |
| *Test Database* | | | 16 | 5 | 5 |
| Create schema | AAA | | 16 | 5 | 5 |
| Populate test database | AAA | | | | |
| | | | | | |
| *Code Reviews* | | | | | |
| Code Review Preparation | AAA,BBB,CCC, DDDD,EEE,FFF | | 14 | 6 | 13 |
| Code Walkthrough Sessions | AAA,BBB,CCC, DDDD,EEE,FFF | | 14 | 6 | 13 |
| | | | | | |
| *Coding* | | | | | |
| Utility Servers | | | | | |
| cDataAccess | AAA,BBB | | 30 | 7 | 7 |
| cError | AAA,BBB | | 15 | 8 | 8 |
| cUI | AAA,BBB | | 15 | 8 | 8 |
| | | | | | |
| Business Servers - Data Centric | | | | | |
| cCompanyDB | CCC,DDD | | 10 | 7 | 7 |
| cPositionsDB | EEE,FFF | | 10 | 7 | 7 |
| cApplicantDB | CCC,DDD | | 10 | 7 | 7 |
| cSkillsDB | EEE,FFF | | 10 | 7 | 7 |
| | | | | | |
| Business Servers - UI Centric | | | | | |
| cCompany | CCC,DDD | | 10 | 9 | 9 |
| cPositions | CCC,DDD | | 5 | 9 | 9 |
| cApplicant | EEE,FFF | | 5 | 9 | 9 |
| cSkills | EEE,FFF | | 10 | 9 | 9 |
| | | | | | |
| UI - Forms | | | | | |
| frmCompany | AAA,BBB | | 20 | 10 | 11 |
| frmPositions | CCC,DDD | | 30 | 10 | 11 |
| frmApplicant | EEE,FFF | | 30 | 10 | 11 |
| frmSkills | AAA,BBB | | 10 | 10 | 11 |
| frmMDIForm | CCC,DDD | | 30 | 10 | 11 |
| frmSplash | EEE,FFF | | 30 | 10 | 11 |
| frmSearch | AAA,BBB | | 10 | 10 | 11 |
| | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| UI - Web | | | | | |
| Applicant.asp | AAA,BBB | | 20 | 12 | 13 |
| Applicantdb.asp | AAA,BBB | | 20 | 12 | 13 |
| | | | | | |
| ***Total Software Development*** | | 80 | 390 | | |
| | | | | | |
| | | | | | |
| | | | | | |
| *Project Management* | | | | | |
| Project Tracking and Status Reporting | DDS | 15 | | 11 | 15 |
| Deliverable Review | DDS | 15 | | 11 | 15 |
| | | | | | |
| *System Test* | | | | | |
| Establish System Test Database | AAA | | 10 | 11 | 11 |
| Create Test Plan | DDS | 10 | | 11 | 11 |
| Create Test Data | AAA | | 10 | 11 | 11 |
| Build Application | BBB | | 10 | 11 | 11 |
| Execute Test | TST | | | 12 | 13 |
| Implement Change Requests | AAA,BBB,CCC, DDDD,EEE,FFF | | 40 | 12 | 13 |
| Sign-off | CLM | | | 13 | 13 |
| | | | | | |
| *User Acceptance Test* | | | | | |
| Establish User Acceptance Test Database | AAA | | 10 | 13 | 13 |
| Create Test Plan | TST | | | 13 | 13 |
| Create Test Data | AAA | | 10 | 13 | 13 |
| Build Application | BBB | | 10 | 13 | 13 |
| Execute Test | TST | | | 14 | 15 |
| Implement Change Requests | AAA,BBB,CCC, DDD,EEE,FFF | | 40 | 14 | 15 |
| Sign-off | CLM | | | 15 | 15 |
| | | | | | |
| *Production Environment* | | | | | |
| Install hardware/software (e.g. IIS,MTS,SQL Sever, Oracle) | EGG | | | 13 | 15 |
| Configure software | EGG | | | 13 | 15 |
| Implement backup/restores | EGG | | | 13 | 15 |
| Implement security strategy | EGG | | | 13 | 15 |
| Create production database | EGG | | | 13 | 15 |
| Initialize database with reference data | EGG | | | 13 | 15 |
| Secure production database | EGG | | | 13 | 15 |
| | | | | | |
| *Deployment* | | | | | |
| Create application setup programs | BBB | | 10 | 13 | 15 |
| Test deployment | BBB | | 10 | 13 | 15 |
| Install application on client machines | EGG | | | | |
| | | | | | |
| ***Total Software Testing and Deployment*** | | 40 | 160 | | |
| | | | | | |
| | | | | | |
| *Conversion* | | | | | |
| Complete detailed mapping of conversion data sources | CCC,DDD | | 20 | 9 | 9 |
| Develop conversion utilities | CCC,DDD | | 80 | 10 | 11 |

| | | | | | |
|---|---|---|---|---|---|
| Test conversion utilities | CCC,DDD | | 40 | 12 | 12 |
| Convert data | CCC,DDD | | 20 | 13 | 13 |
| Verify converted data | TST | | | 13 | 13 |
| | | | | | |
| ***Total Conversion of Data to New Format*** | | 0 | 160 | | |
| | | | | | |
| | | | | | |
| *Project Management* | | | | | |
| Project Tracking and Status Reporting | DDS | 20 | | 16 | 20 |
| Deliverable Review | DDS | 20 | | 16 | 20 |
| | | | | | |
| *Training* | | | | | |
| Create training plan | EEE,FFF | | 20 | 16 | 16 |
| Develop training materials | EEE,FFF | | 80 | 17 | 18 |
| Schedule training | EEE,FFF | | 20 | 18 | 18 |
| Train users | EEE,FFF | | 40 | 19 | 20 |
| | | | | | |
| ***Total User Training and Support*** | | 40 | 160 | | |
| | | | | | |
| | Initials: | | | | |
| | DDS-Project Manager | | | | |
| | AAA-Developer | | | | |
| | BBB-Developer | | | | |
| | CCC-Developer | | | | |
| | DDD-Developer | | | | |
| | EEE-Developer | | | | |
| | FFF-Developer | | | | |
| | TST-Client Tester | | | | |
| | CLM-Client Manager | | | | |
| | EGG-Client Technician | | | | |

# 4. Coding Conventions

## *Database*

- Each field will be prefixed with the table name.
- Table names and field names will NOT contain spaces.
- Referential integrity and business rules (i.e. required fields, duplicates allowed, etc.) will be defined within Access.

  *Note: Although this is contrary to the advice given in Doing Objects in Visual Basic 6, many large organizations with established Data Base Administration (DBA) organizations require referential integrity and business rules to be included in the database. In theory, all access to the data should be through the business objects, however, in practice, this unfortunately is not usually the case. Therefore, until full commitment to the object-oriented approach is established throughout an organization, referential integrity and business rules will likely be implemented in the database as well as the application. The disadvantage to this approach is increased maintenance, because as business rules*

*change, the changes will have to be implemented in the business objects as well as the database.*

- ID fields will be type Long.
- It is generally preferred that the ID field be used as foreign keys (reference fields). Foreign Keys will use the name of the field in the referenced table.

## *Visual Basic*

### General Coding

- All modules will begin with the following:

```
' Class/Form/Module Name:
' Author:
' Date:
' Description:
```

- All routines (subroutines, functions and property procedures) begin with a brief comment describing the functional characteristics of the routine as well as the parameters passed to and returned from the routine:

```
' Purpose:
' Parameters:
' Returns:
' Sets:
```

- All modules include Option Explicit to require variable declarations.
- Routines are ordered as follows:

  Property Procedures
  Initialize and Terminate Event Procedures
  Public sub and function procedures (in alphabetic order)
  Private sub and function procedures (in alphabetic order)

- Use indents to show nesting program structures.
- Declare all necessary variables at the top of each procedure.
- Place error handling in each procedure.
- Use constants rather than magic numbers and literals within the code.
- Strings should be placed in the resource file and not hard coded.
- Use a "." operator between a form and a control rather than the "!".
- Case statements should include a Case Else.
- Use the line continuation character for long lines.
- Don't put multiple statements on one line.
- Explicitly state the scope of variables and procedures (i.e. Public or Private)

### Naming Conventions

**Object Naming**

| Prefix | Object Type |
|---|---|
| ani | Animated button |
| cbo | Combo box |
| chk | Check box |
| cmd | 3D command button |
| col | Collection |
| ctl | Control |
| dat | Data |
| db | Database |
| dbcbo | Data-bound combo box |
| dbgrd | Data-bound gid |
| dblst | Data-bound list box |
| dbc | Data combo |
| dgd | Data grid |
| dbl | Data list |
| drp | Date combo box |
| dtp | Date picker |
| dir | Dir list box |
| dlg | Common dialog |
| drv | Drive list box |
| fil | File list box |
| frm | Form |
| fra | Frame |
| gau | Gauge |
| gra | Graph |
| grd | Grid |
| flex | Hierarchical flexgrid |
| hsb | Horizontal scroll bar |
| img | Image |
| ils | Image List |
| key | Keyboard key status |
| lbl | Label |
| lin | Line |
| lst | List box |
| lsv | List View |
| lwchk | Lightweight check box |
| lwcbo | Lightweight combo box |
| lwcmd | Lightweight command button |
| lwfra | Lightweight frame |
| lwhsb | Lightweight horizontal scroll bar |
| lwlst | Lightweight list box |
| lwopt | Lightweight option button |
| lwtxt | Lightweight text box |
| lwvsb | Lightweight vertical scroll bar |
| mnu | Menu |
| mvw | Month view |
| opt | Option button |
| pic | Picture box |
| pnl | 3-D panel |
| prg | Progress bar |
| ps | Prepared statement |
| rpt | Report |

| | |
|---|---|
| rtf | Rich Text Box |
| rs | Result set |
| shp | Shape |
| sld | Slider |
| spn | Spin control |
| sta | Status bar |
| sys | SysInfo |
| tab | Tab Strip |
| tlb | Toolbar |
| tmr | Timer |
| tre | Treeview |
| txt | Text box |
| vsb | Vertical scroll bar |

**Menu Naming**
Menus should be named similarly to their caption. Each item in the menu should be in a control array that is named using the menu name and the suffix Item (EditItem, FileItem, etc.)

**Module Naming**
The files for Forms, Interfaces, and Classes and Modules should be named for what objects they work with. The objects should be named with the following prefixes

| **Prefix** | **Object Type** |
|---|---|
| frm | Form |
| C | Classes |
| I | Interfaces |
| M | Modules |

**Object Method Naming**
Method names are generally denoted in Title Case with a Verb or Verb/Noun combination. Methods, which perform the same function for different objects, should be named consistently.

**Examples:**

FillForm
CheckRow
Enable
Notify
Display
Add
Remove

**Variable Naming**
A variable name consists of three components:

The **scope** prefix:

| Prefix | Scope |
|---|---|
| g | Global |
| m | module-level |
| st | Static |
| (none) | local or parameters |

The **data type** prefix:

| Prefix | Data Type |
|---|---|
| b | Boolean |
| bt | Byte |
| col | Collection |
| d | Double - 64 bit signed quantity |
| dt | Date+Time |
| e | Error |
| f | Float/single - 32-bit signed floating point |
| h | Handle |
| i | Integer |
| l | Long - 32 bit signed quantity |
| obj | Object |
| s | String |
| v | Variant |
| udt | User defined type |

**The variable purpose:**
A logical description of the variable usually in Title Case. Should be long enough to describe its purpose.

**Examples:**

| | |
|---|---|
| m_objCurrency | Module level object variable |
| sFieldName | Local string variable |
| bIsDirty | Local boolean variable |
| m_bIsDirty | Module level boolean variable |
| g_objApp | Global object variable |

**Constant Naming**
Constants names are uppercase with "_" between words.

Constants used to identify indexes in control arrays should be prefixed with the appropriate prefix for the type of control it represents.

**Examples:**
cboCURRENCY_INVERTED
txtCURRENCY_STATUS
ERR_INVALID_NULL
MV_FIRST

# 5. Quality Assurance

Code reviews are to take place with the development team. During this review, each of the following questions will be discussed:

- Is a consistent user-friendly interface applied?
- Did the user-interface appropriately enable or disable functionality based on the state of the application?
- Is a standard module used for duplicate constants, sub procedures, functions, etc.?
- Are standard object and variable naming conventions consistently followed throughout the application?
- Is program logic put in place to ensure invalid data was not passed to the database?
- Does the application address the three major sections (Company, Position, Applicant)?
- Does the application use an MDI environment?
- Does the application use menus and a toolbar?
- Does the application use access keys whenever possible?
- Were unexpected errors encountered during acceptance testing?