Palestine Polytechnic University

College of Engineering & Technology
Electrical & Computer Engineering Department

Software Project

Speech-Enabled Web Application
"Case Study: Arab Bank Website"

Project Team
Ahmad Al-Manasra **"Mohammed Sharaf" Al-Zabadi**

Project Supervisor
Eng. Murad Abu Sbeih

Hebron-Palestine

June, 2004

Palestine Polytechnic University

Hebron-Palestine

College of Engineering & Technology
Electrical & Computer Engineering Department

Software Project

Speech-Enabled Web Application
"Case Study: Arab Bank Website"

Project Team

Ahmad Al-Manasra **"Mohammed Sharaf" Al-Zabadi**

Project Supervisor

Eng. Murad Abu Sbeih

In accordance with the Engineering & Technology College's System, and the supervision of the project supervisor, and the acceptance of all examining committee members, this project has been submitted to the Department of Electrical & Computer Engineering in the College of Engineering and Technology in partial fulfillment of the requirements of Department for the degree of Bachelor of Science in Engineering, major Computer Systems Engineering.

Project Supervisor

…………………….

Examination Committee

……………………. **…………………….** **…………………….**

Department Chairman

…………………….

ABSTRACT

Our project is designing a web application that can be interacted with visually i.e. using mouse and keyboard or by voice. Such applications are called Multimodal or Tab-And-Talk Applications. This means that when user wants to speak to the web application he needs to click a button and then speaks.

ملخص

يهدف هذا المشروع إلى تصميم صفحات انترنت يمكن التعامل معها باستخدام لوحة المفاتيح والفأرة أو باستخدام الصوت. في هذا النوع من تطبيقات الإنترنت يحتاج المستخدم إلى الضغط على زر معين ثم يبدأ بالتحدث الى الحاسوب.

DEDICATION

To my mother and father

To rose of my life, to the SunRise

Mohammed sharaf

To my family and friends

Ahmad

ACKNOWLEDGMENT

First of all, we would like to express our thanks to our university (Palestine Polytechnic University) that gave us this opportunity to achieve this project. We also thank The Collage of Engineering & Technology and The Department of Electrical & Computer Engineering for their valuable assistance.

We also wish to record our appreciation of the supervision, guidance, and support of our project's supervisor Eng. Murad Abu Sbeih.

Table of Contents

List of Tables

List of Figures

Chapter One
Introduction

Chapter One
Introduction

1.1 Preface

Nowadays technology has obviously entered our lives, and has become an essential part of it in a way that no one can keep up without it, mainly in communications and information technology.

If we take a look at communication technology, we will find that Internet has been widely used by individuals and organizations. Individuals use Internet to get information and to do some of their work, organizations use it to offer services to their customers and for society, so they can access the organization resources and get benefit from its services remotely.

A new technology has been arised in the last few years, which is the speech technology, this technology enables the user to speak to the computer, one example for speech technology is the Microsoft office XP, users in USA, JAPAN, and CHINA can dictate texts to Microsoft word and PowerPoint documents.

Users can also dictate commands and manipulate menus by speaking. For many users dictating is quicker and easier than using keyboard. Office XP can speak back too. For example Microsoft Excel can read text back to the user as the user enters it into cells saving the trouble of checking back and forth from screen to paper.

Another speech technology has been introduced by Microsoft, is speech application SDK, this application is a member of Visual Studio.Net, it is used to make the ASP.NET application voice enabled, so the interaction between the user and the Internet application will not only be visually but it also will be using voice which called multimodal.

In this project we are going to design a website for a bank. The database of the bank will be connected with the internet using voice enabled ASP.NET i.e. using speech application SDK. This speech enabled web application will allow the customer to use the services available using both voice and visual interaction.

The customer can interact with web pages visually (using mouse and keyboard) for input or output data needed, on the other hand he may click a button and then speak to the computer.

This will add naturalness and power of speech interaction to the ordinary Internet applications.

1.2 System Objectives

1- Allow customers to contact with the bank remotely.
2- Apply the voice recognition technology on the ordinary web application programming languages.
3- Make the communication between the bank and the customer easier, more comfortable and more interesting.
4- Add naturalness to the interaction between the user and the system.
5- Design a speech application that can deal with different responses expected from the users.

1.3 Previous Work / Previous System

Current web based systems are traditional, we don’t mean old but it is like any other Internet application, the only way to interact with it is visually, so when the user enters to the bank web site, he will use only the keyboard or the mouse, then he can enter the data he wants or ask about any information needed.

Table 1.1 compares between the current system and the proposed system

Aspect of Comparison	Traditional System	Proposed System
Efficiency	Lower	Higher
Time	Needs more time	Less time

Efforts	Needs more efforts	Less efforts
Accuracy	Same	Same

1.4 Main Points of Contribution

The main noticeable idea in this system is the interaction with the computer by voice, the user may tell the computer what he wants and then the computer responds, so the user is free to use the mouse and keyboard, or his voice to perform tasks. Also, the used technology is new and few web sites are seen to use it.

1.5 Report Outline

We divided our report into seven chapters. The subjects included in each chapter are shown in the following sub-sections.

Chapter One

In this chapter we talk about our system in a general introduction. After that we mentioned the objectives of the system and a comparison between it and the previous systems. Also the main points of contribution and a theoretical background have been included in this chapter.

Chapter Two

This chapter talks about the team project and the supervisor. Besides it discusses the system risks, the hardware and the software requirements, the cost of the system, and the system schedule.

Chapter Three

This chapter includes a detailed description of the functional and the non-functional requirements of the system.

Chapter Four

This chapter includes the data needed for the system. It also discusses the outcomes and design models of the system.

Chapter Five

This chapter shows the system database, the steps needed to implement the system, the grammars used in the system, and the system security aspects.

Chapter Six

This chapter involves the steps followed to test the system. This includes Unit Testing, Sub-System Testing and System Testing. It also talks about Acceptance Testing.

Chapter Seven

This chapter contains the conclusions gained after finishing our project and some guidelines for future work.

1.6 Theoretical Background

1.6.1 Introduction to Computer Speech Technology

In the mid to late 1990s, personal computers started to become powerful enough to enable users to speak to them and for the computers to speak back. While speech technology is still far from delivering natural, unstructured conversations with computers, it currently is delivering some very real benefits in real applications. For example:

- Many large companies have started adding speech recognition to their Interactive Voice Recognition (IVR) systems. Just by phoning a number and speaking, users can buy and sell stocks from a brokerage firm, check flight information with an airline company, or order goods from a retail store. The systems respond using a combination of prerecorded prompts and an artificially generated voice.

1.6.2 Speech Recognition

1.6.2.1 Introduction to Speech Recognition

Speech recognition (SR) is the process of converting spoken language into printed text. Speech recognition, also called speech-to-text recognition, involves:

1. Capturing and digitizing the sound waves produced by a human speaker.
2. Converting the digitized sound waves into basic units of language sounds or phonemes.
3. Constructing words from the phonemes.
4. Analyzing the context in which the words appear to ensure correct spelling for words that sound alike (such as write and right).

Figure 1.1 below illustrates a general overview of the process.[7]

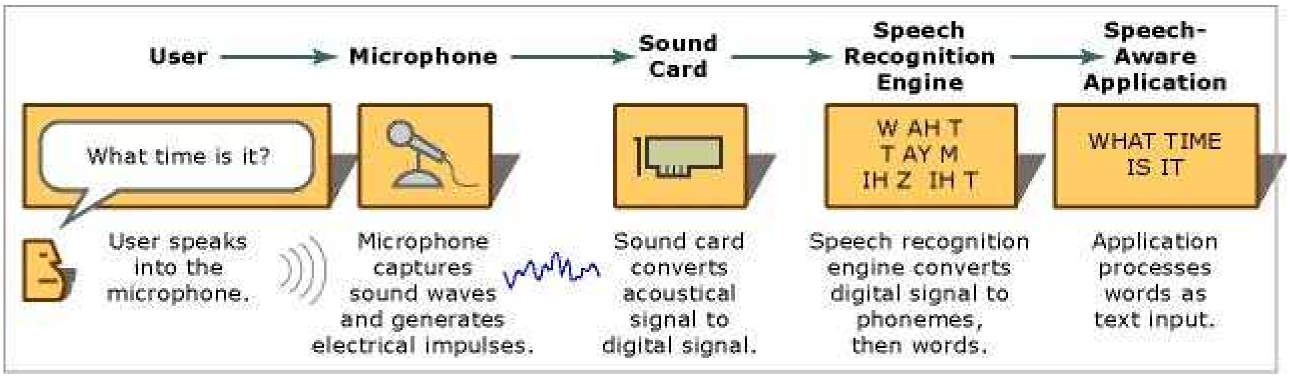

User
Microphone
Sound Card
Speech Recognition Engine
Speech-Aware Application
What time is it?
W AH T
T AY M
IH Z IH T
WHAT TIME IS IT
User speaks into the microphone.
Microphone captures sound waves and generates electrical impulses.
Sound card converts acoustical signal to digital signal.
Speech recognition engine converts digital signal to phonemes, then words.
Application processes words as text input.

Figure 1.1: The process of voice recognition

Recognizers (also known as speech recognition engines) are the software drivers that convert the acoustical signal to a digital signal and deliver recognized speech as text to an application. Most recognizers support continuous speech recognition, meaning that users can speak naturally into a microphone at the speed of most conversations. Isolated or discrete speech recognizers require the user to pause after each word, and are currently being replaced by continuous speech engines.

Continuous speech recognition engines currently support two modes of speech recognition:

- Dictation, in which the user enters data by reading directly to the computer.
- Command and control, in which the user initiates actions by speaking commands or asking questions.

Using dictation mode, users can dictate memos, letters, and e-mail messages, as well as enter data. The size of the recognizer's grammar limits the possibilities of what can be recognized. Most recognizers that support dictation mode are speaker-dependent, meaning that accuracy varies depending on the user's speaking patterns and accent. To ensure the most accurate recognition, the application must create or access a speaker profile that contains information about the user's speech patterns.

Using command and control mode, users can speak commands that control the functions of an application. Implementing command and control mode is the easiest way for developers to integrate a speech interface into an existing application because developers can limit the content of the recognition grammar to the available commands. This limitation has several advantages:

- It produces better accuracy and performance rates for command and control speech recognition than for continuous dictation speech recognition, because a continuous speech recognition engine must encompass nearly an entire language dictionary.
- It reduces the processing overhead that the application requires.

- It also enables speaker-independent processing, eliminating the need for speaker profiles or "training" of the recognizer.

1.6.2.2 Speech Recognition Using Speech Application SDK

Speech recognition using the Microsoft Speech Application SDK Version 1.0 Beta 3 (SASDK) is a process with two distinct parts. The first part involves converting speech to text as described earlier. The second part involves semantic analysis of the resulting recognition text in order to determine its meaning. The recognizer iteratively compares the recognition text to the rules in the application's grammar. When the recognizer matches recognized text to a series of rules in a grammar, the recognizer produces an XML output stream, using Semantic Markup Language (SML), to represent semantic output. The semantic output contains recognition confidence values, recognized text, and can also contain semantic values that the developer assigns using semantic interpretation markup. Developers use the information in the SML output to infer the meaning of what the user said.

1.6.2.3 What Speech Recognition Adds to Applications

Using speech recognition technology, developers can produce applications that feature:

- Hands-free computing, either as an alternative to the keyboard, or to enable users to use the application in environments where a keyboard is impractical (as with small mobile devices, AutoPCs, or mobile phones).

- A more human-like computer interface, making educational and entertainment applications seem more friendly and realistic.

- Voice responses to message boxes and wizard screens.

- Streamlined access to application controls and large lists, enabling a user to speak any item from a list, or any command from a large set of commands without navigating through several dialog boxes or cascading menus.

- Context-sensitive dialogs between the user and the computer in which the computer's response depends on the user's input. For example, imagine a travel services application asking a user, "What do you want to do?", and the user replying, "I want to book a flight from New York to Boston." In this case, the application could respond by asking whether the user wants to book a flight departing from La Guardia airport, or JFK airport. After receiving this information, the application could continue by asking what day and time the user wants to leave. On the other hand, if the user wants to book a flight departing from a city with only one airport, the application would not need to clarify the departure airport, and could move immediately to asking about a departure day and time.

1.6.2.4 Potential Applications for Speech Recognition

The specific uses of speech recognition technology depend on the application type. Some of the types of applications that are good candidates for implementing speech recognition include:

- Telephony

Speech recognition plays a critical role in telephony applications. Many telephony applications require users to press telephone keypad numbers in order to select from a list of items. Pressing keypad numbers can be inconvenient for users with cell telephones or handset telephones. Speech recognition can make placing orders, selecting items, and providing other information over the phone more natural.

- Data Entry

Speech recognition can significantly improve the speed of data entry of numbers and items selected from a small list (less then 100 items). Some recognizers can even handle spelling fairly well. If a spreadsheet or database application has fields with mutually exclusive data types (for example, one field that recognizes *male* or *female,* another that recognizes age, and a third that recognizes a city name), developers can program the application to automatically populate the correct data fields in the spreadsheet or database if the user provides all of the requested pieces of information in a single utterance.

- Games and Edutainment

Speech recognition enhances the realism and fun in many computer game and edutainment applications by enabling users to talk to on-screen characters as if they were talking to another person.

- Document Editing

In dictation mode, speech recognition can enable users to dictate entire documents into a word processor without typing. In command and control mode, speech recognition can enable users to modify document formatting or change views without using the mouse or keyboard. For example, a word processing application can provide commands like "bold", "italic", "change to Times New Roman font", "use bullet list text style," and "use 18 point type", while a graphic manipulation application can provide commands like "select eraser" or "choose a wider brush."

1.6.3 Text-To-Speech Synthesis

1.6.3.1 Introduction to Text-To-Speech Synthesis

Text-to-speech (TTS) synthesis is the process of converting text into spoken language. This process involves:

1. Breaking down the words of the text into phonemes.
2. Analyzing the input for occurrences of text that require conversion to symbols, such as numbers, currency amounts, and punctuation (a process known as inverse text normalization, or ITN).
3. Generating the digital audio for playback.

Figure 1.2 illustrates a general overview of the process. [7]

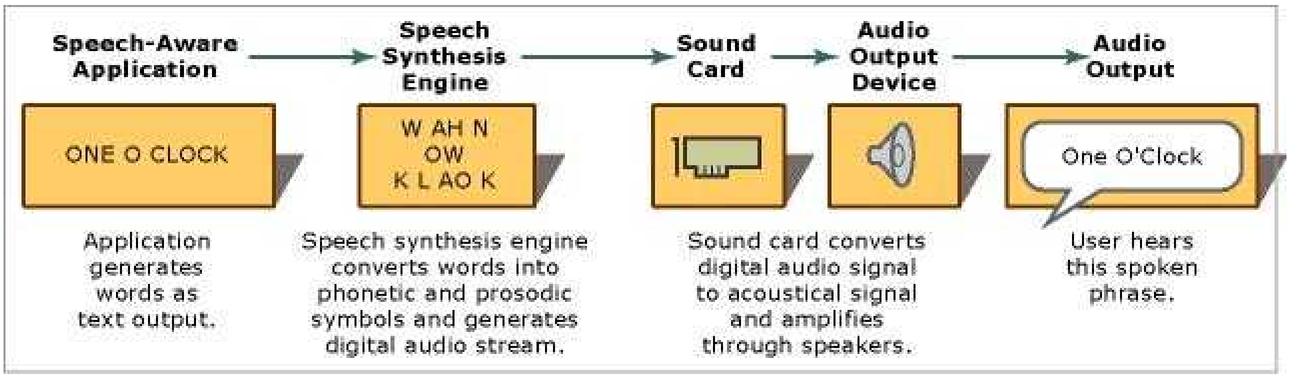

Speech-Aware Application
Speech Synthesis Engine
Sound Card
Audio Output Device
Audio Output
ONE O CLOCK
W AH N OW K L AO K
One O'Clock
Application generates words as text output.
Speech synthesis engine converts words into phonetic and prosodic symbols and generates digital audio stream.
Sound card converts digital audio signal to acoustical signal and amplifies through speakers.
User hears this spoken phrase.

Figure 1.2 : Process of Text-to-speech

Text-to-speech (TTS) synthesis engines (also referred to as text-to-speech voices) are the software components that perform speech synthesis, handling the complexity of converting text to spoken language. TTS engines use one of two approaches:

- One is to generate artificial sounds similar to those created by human vocal cords and apply various filters to simulate throat length, mouth cavity shape, lip shape, and tongue position.

- The other is to work with a database containing numerous prompts recorded by a speaker, extract segments of speech from various prompts, and then concatenate those segments to create a prompt that was never spoken by the original speaker.

For example, a speaker can record the two statements "Your flight leaves at two P.M. tomorrow" and "That item costs four dollars." The developer can extract the segments "Your flight leaves at", and "P.M. tomorrow" from the first statement, and "four" from the second statement. By concatenating these segments, the prompt engine can generate the new statement "Your flight leaves at four P.M. tomorrow."

Although easy to understand, a prompt produced by these approaches (artificial synthesis and segment concatenation) tends to sound less human than an individual, continuous recording of the same prompt spoken by a human speaker. Nevertheless, text-to-speech applications can be the better alternative in situations where individual audio recordings of every individual prompt are inadequate or impractical.

Generally, developers consider using TTS synthesis when:

- Audio recordings are too large to store on disk, or are prohibitively expensive to record.
- The prompts that the application plays consist of only short phrases.
- The developer cannot predict what responses users will require from the application (such as requests to read e-mail over the telephone)
- The number of alternate responses required makes recording and storing prompts unmanageable or prohibitively expensive.
- The user prefers or requires audible feedback or notification from the application. For example, a user may prefer to use TTS to perform audible proofreading of text and numbers in order to catch typographical errors missed by visual proofreading.

1.6.3.2 Potential Applications for Speech Synthesis

The specific uses of TTS technology depend on the application type. Some of the applications that are good candidates for implementing TTS include:

- ➢ Telephony

Text-to-speech synthesis plays a critical role in telephony applications. Because telephony applications have no visual interface, using TTS synthesis is a valuable method for verifying customer selections. TTS can also be the preferred method for delivering information that users request, especially when the range of the information requested derives from a large set of possible values. Stock price information is a good example. Although it is possible to record all of the numbers that are likely to be part of a stock price, doing so is time-consuming and expensive, making TTS a faster, less expensive alternative.

- Data Entry

Developers can implement text-to-speech reading of data values as users enter data in a spreadsheet or database application as a means of verifying correct entry of information that can be tedious to check, such as phone numbers, addresses, monetary values, and identification numbers.

- Games and Edutainment

Text-to-speech synthesis allows the characters in an application to talk to the user instead of merely displaying speech balloons. Even if it is possible to use digital recordings of speech, using TTS instead of recordings can be preferable in certain cases. For example, the less-than-human quality of artificially synthesized speech makes it ideal for characters whose voices are robotic or alien.

In addition, TTS can be generally useful for application prototyping. In some cases TTS may even be the only practical alternative. For example, if application development schedules do not allow enough time to record all of the prompts that an application requires, the application developer may have no alternative to using TTS.

1.6.4 Microsoft Speech Application SDK Version 1.0 Beta 3 (SASDK)

The Microsoft Speech Application SDK Version 1.0 Beta 3 (SASDK) is a set of developer tools, samples, and documents that allow ASP.NET Web developers to create, debug, and deploy speech-enabled ASP.NET applications. The speech application authoring environment is seamlessly integrated into Microsoft Visual Studio .NET 2003 allowing the developer to leverage the strengths of the Visual

Studio .NET 2003 development environment such as toolboxes, graphical development environments, and multiple views.

1.6.4.1 The main elements of SASDK

The main elements in the SASDK are:

- A set of ASP.NET Speech Controls (Speech Controls) that enable speech input and output in ASP.NET applications by generating HTML + SALT markup for telephone (voice-only) and multimodal browsers.
- Four Visual Studio .NET 2003 add-on tools that developers use to speech-enable an application with few speech-specific knowledge requirements:

    1. A tool for creating and editing Speech Controls
    2. Speech Grammar Editor for creating and editing speech recognition grammars
    3. Speech Prompt Editor for creating and editing prerecorded voice output
    4. Speech Debugging Console

- Speech Add-in for Microsoft Internet Explorer that allows developers to run and test their speech-enabled Web applications.

1.6.4.2 Speech Applications Modes

Speech applications that are built using the Microsoft Speech Application SDK Version 1.0 Beta 3 (SASDK) run in either telephony (voice-only) or multimodal mode. The most dramatic difference between voice-only and multimodal applications is in the way users interact with them.

For voice-only applications, users typically call the application using a telephone. Users respond to verbal prompts by providing spoken answers, or by pressing keys on the phone keypad. An example might be a customer calling an airline to get arrival and departure time updates.

Multimodal applications display a graphical user interface (GUI). Users open multimodal applications by entering a Web address in Microsoft Internet Explorer. Users then speak answers in response to visual or graphical prompts on a Web page. To speak to a multimodal application, users typically click and a control on a page, speak an answer. When the speech recognition engine recognizes the user's speech, it sends the recognized text to the application, and the text appears in a text box, a drop-down list, or some other control.

Chapter Two

Project Planning

Chapter Two
Project Planning

2.1 Preface

* The project team consists of two students:
- Ahmad Almanasra.
- Mohammad Alzabadi.

* Project supervisor:
- Eng. Morad Abu Sbeih.

Both of the two students started working together in collecting data and information necessary for the project, and then they worked in analyzing the requirements and the design. After that, they implemented and tested the project, also wrote the documentation together.

2.2 Risk Analysis

The system risks are:

- ❖ Communication between new system and users.
- ❖ The system miss recognize some speech inputs.

- ❖ The system may not work well if there is a noise around.

The first point means that users of the new system may not accept it at first, because they don’t know how to use it, or because they don’t speak English. Using the provided help the user can get over this risk and use the system normally.

The second point is that users don’t talk in the same way and the spelling of some words may be different from one user to another, so the system may fail to recognize the spoken commands. To solve this problem the speech engines may be made clever to deal with different patterns of speech.

The last point means if there is a noise around the user it will affect the system because it may respond to another voice or it will misunderstand what has been said. To avoid this risk the user has to be away of noise when he want to use the system.

2.3 Hard ware and software requirements

2.3.1 Hardware development resources

We need the following preferred hardware resources to develop our system:

- Two PCs with the following specifications:

- Pentium IV processor.
- 512 MB RAM.
- 40 GB hard disk.
- 17 SVGA Monitor.
- Keyboard + Mouse.
- Microphone.
- Speakers.
- 56 Kbps dial-up modem.
- Internet connection.

2.3.2 Hardware Operational Resources

Hardware needed for implementation must be stable and powerful and it is categorized as the following:

- Pentium IV processor.
- 512 MB RAM
- 40 GB hard disk.
- 17 SVGA monitor.
- Keyboard + Mouse.
- 56 Kbps dial-up modem.
- 10/100 Mbps Ethernet network adapter card.
- Internet connection.
- Microphone.
- Speakers.

2.3.3 Software development resources:

Each PC must have the following installed software:

- Window XP.
- Visual studio.NET.
- Speech application SDK V.1 Beta 3.
- MS SQL server.
- MS office XP/2000.
- IIS web server.
- Internet Explorer with Add-in tools.

2.3.4 Cost Estimation

- **Hardware costs**

  2* PC Pentium IV : 2*750$=1500$.

- **Software Costs:**

Windows XP	=	500$
Visual Studio.NET	=	630$
MS SQL Server	=	350$

SDK V.1Beta 3 = 500$

Total Software Costs = 300+630+350+500=1780$

The total costs are:

Hardware Costs + Software Costs = 1500 + 1780 = 3280$

2.4 Work Activities

Developers started acquiring information about the overall system and its consideration by asking banks and programmers and also fetching for data from the internet and books.

After having a general overview about how the system should be , the following activities emerged to be done in a scheduled time:

- Project planning, scheduling and definition.
- Requirements analysis and specifications.
- System design.
- Implementation.
- Testing.
- Documentation.

2.5 Project scheduling

The system implementation time is about 15 weeks. These weeks are organized and arranged in a manner to not exceed the dead line of each task. Table 2.1 depicts the managed time weekly:

Table 2.1: Project Scheduling

Weeks **Tasks**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
System Planning	▨	▨	▨	▨	▨										
Requirements Analysis			▨	▨	▨	▨	▨								
System Design					▨	▨	▨	▨	▨	▨					
System Implementation						▨	▨	▨	▨	▨	▨	▨	▨		
System Testing											▨	▨	▨	▨	
Documentation	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

Chapter Three

System Requirements

3.2 Non-Functional Requirements

3.1 Functional Requirements

Chapter Three
System Requirements

The system requirements are categorized into two main parts, which are:

- Functional Requirements
- Non-functional Requirements

3.1 Functional Requirements

We will illustrate the functional requirements through this section. Functional requirements definition and functional requirements specification will be considered.

3.1.1 Functional Requirements Definition

The functional requirements involve accomplishing the followings:

1. Allow the customer to check his balance.
2. Allow the customer to display the account state.
3. Allow the customer to order a cheque book.
4. Allow the customer to transfer money.
5. Allow the customer to view the prices of currencies.

3.1.2 Functional Requirements Specification

3.1.2.1 Allow the customer to check his balance

Software Project* /Speech Enabled Web Application / *3.1.2.1

Function: Allow the customer to check his balance

Description: This requirement enables the user to check the balance available in his account. The customer's Account Number, type and currency are input either using voice or keyboard.

Inputs: Account number, Password, Account type, and Account currency.

Source: Account number, Password, Account type, and Account currency are input by the customer.

Outputs: The system will show the customer the balance available. If any error appears, an error message emerges to justify that error.

Destination: The Bank database.

Requires : Visiting the Bank home page first and select a service, then passing the login web page by entering a valid account number and password.

Pre-Condition: The customer has to visit the home page of the Bank then redirected to the required page according to the chosen service. The customer also has to provide a valid account number and password in order to use the service.

Post-Condition: Confirmation or Error message appears.

Side-effects: none.

Definition: ***Software Project* /Speech Enabled Web Application / *3.1.2.1***

3.1.2.2 Allow the customer to display the account state

Software Project /*Speech Enabled Web Application *& 3.1.2.2

Function: Allow the customer to display the account state.

Description: this requirement enables the customer to display the account state between two dates. This includes any transaction that has been performed on the account in the period between the two entered dates.

Inputs: Account number, Password, Account type, Account currency, First date, and Second date.

Source: Account number, Password, Account type, Account currency, First date, and Second date are input by the customer.

Outputs: The system will show the transactions that have been performed on the account. For each transaction the system shows the amount of money used in the transaction, the date of the transaction occurrence, balance available after transaction and any necessary details. If any error occurs, an error message emerges to justify that error.

Destination: The Bank database.

Requires : Visiting the Bank home page first and select a service, then passing the login web page by entering a valid account number and password.

Pre-Condition: The customer has to visit the home page of the Bank then redirected to the required page according to the chosen service. The customer also has to provide a valid account number and password in order to use the service.

Post-Condition: Confirmation or Error message appears.

Side-effects: none.

Definition: ***Software Project /*Speech Enabled Web Application */ 3.1.2.2***

3.1.2.3 Allow the customer to order a cheque book

Software Project /*Speech Enabled Web Application */ 3.1.2.3

Function: Allow the customer to order a cheque book.

Description: this requirement enables the customer to order a cheque book. This includes choosing the specifications of the cheque book such as: language, number of pages and delivery method.

Inputs: Account number, Account type, Account currency, Password, Language of cheque, Number of pages, and Delivery method.

Source: Account number, Account type, Account currency, Password, Language of cheque, Number of pages, and Delivery method are input by the customer.

Outputs: The system will send the order and store it in the bank data base to be served later. If any error appears, an error message emerges to justify that error.

Destination: The Bank database page.

Requires : Visiting the Bank home page first and select a service, then passing the login web page by entering a valid account number and password.

Pre-Condition: The customer has to visit the home page of the Bank then redirected to the required page according to the chosen service. The customer also has to provide a valid account number and password in order to use the service.

Post-Condition: Confirmation or Error message appears.

Side-effects: none.

Definition: ***Software Project /*Speech Enabled Web Application */ 3.1.2.3***

3.1.2.4 Allow the customer to transfer money

Software Project /**Speech Enabled Web Application */ 3.1.2.4***

Function: Allow the customer to transfer money

Description: this requirement enables the customer to transfer money from his account to another account in the same bank i.e. internal transfer.

Inputs: Source account number, Source account type, Source account currency, Password, Destination account number, Destination account type, Destination account currency, and Amount of money to be transferred.

Source: Source account number, Source account type, Source account currency, Password, Destination account number, Destination account type, Destination account currency, and Amount of money to be transferred are input by the customer.

Outputs: The system will transform the specified amount of money from the source account to the destination account. If any error appears, an error message emerges to justify that error.

Destination: The Bank Database.

Requires : Visiting the Bank home page first and select a service, then passing the login web page by entering a valid account number and password.

Pre-Condition: The customer has to visit the home page of the Bank then redirected to the required page according to the chosen service. The customer also has to provide a valid account number and password in order to use the service.

Post-Condition: Confirmation or Error message appears.

Side-effects: none.

Definition: ***Software Project /*****Speech Enabled Web Application */ 3.1.2.4***

3.1.2.5 Allow the customer to view The Exchange Rates

Software Project /*Speech Enabled Web Application */ 3.1.2.5

Function: Allow the customer to view the prices of currencies

Description: this requirement enables the customer to know the price of any currency in respect to any other currency

Inputs: Account number, Password, Source currency, and Destination currency.

Source: Account number, Password, Source currency, and Destination currency are input by the customer.

Outputs: The system will show the price of exchange the two entered currencies. If any error appears, an error message emerges to justify that error.

Destination: The Bank Database.

Requires : Visiting the Bank home page first and select a service, then passing the login web page by entering a valid account number and password.

Pre-Condition: The customer has to visit the home page of the Bank then redirected to the required page according to the chosen service. The customer also has to provide a valid account number and password in order to use the service.

Post-Condition: Confirmation or Error message appears.

Side-effects: none.

Definition: ***Software Project /*Speech Enabled Web Application */ 3.1.2.5***

3.2 Non-Functional Requirements

When working with web application, security requirements became an essential point in designing and developing such applications. Since the project consists of designing a web application for a bank, security must be implemented as much as possible else the bank database will be subjected to hackers.

Beside security, there are other Non-Functional requirements such as **Friendly User Interface** and **Data Encryption** requirements needed to be included in the system.

3.2.1 Security

Security is the first point that is needed for both of the bank and the customer. Its not acceptable to develop a web application for a bank without taking into consideration the security of the bank database.

To secure our system we've implemented the following tools:

- MS SQL Server 2000 Stored Procedure.
- Forms-Based Authentication.

The following are the general aspects of the system security:

1. For each customer only one account number exists. But the customer may have different types of accounts with different currencies under the same account number.

2. No customer can use the services available on the bank website without having an account number and password.
3. No customer can access the system services by writing their URL directly.
4. After passing the login page to the service page no, customer can apply the service to another account number.

3.2.2 Friendly User Interface

Visual Studio .NET is a power full tool for developing web-based systems. It provides all the required components to develop a friendly user interface. Menus, Drop down lists, Data grids, Buttons, Labels, Hyperlinks and other graphical components were incorporated among the system to grant intelligible interface.

3.2.3 Data encryption Through Certificate Server

The system uses SSL (Secure Soket Layer) protocol to establish a secure connection between the Client (Web Browser) and the Server (IIS). This secure communication guarantees that no one other than the client and server can understand the data exchange between them. i.e. data encrypted and then goes between client and server.

Chapter Four
System Design and Analysis

Chapter Four
System Design and Analysis

4.1 Introduction

This chapter shows the system models. In addition, it demonstrates the relations between all system parts. System design is how to control the data flow between modules. Each subsystem depends on its input to control data flow in order to generate output.

4.2 Data Needed for the System

After doing some kinds of surveying and after studying the current systems of the banks to manage the customers', activities we deduced the data that are needed to operate the system.

The data needed to operate the system are categorized as follows:

4.2.1 Bank Main Activities:

This envolves the activities that the bank offerrs to its customers through its web site, and they are:

- Check Balance.
- Transfer Money.
- Currency Prices.
- Account State.
- Cheque Book Order.

4.2.2 The Customer Data:

This involves information that identifies a customer, such as:

- Customer’s ID.
- The customer’s account number.
- Provision of the customer first and last name.
- The customer’s password.
- The customer’s address.
- Customer’s telephone number.
- Customer’s mobile number .
- Customer’s FAX number .
- The customer’s Email.
- Some notes that will be neede in future.

4.2.3 The Account Data:

This involves information about the accouunts of customers and they are categorized into two parts:

- The account data.
- The account type.

4.2.3.1 Account Data:

The data about a certain account, such as:

- ❖ Account nubmer.
- ❖ The account type number.
- ❖ The currency number.
- ❖ The account state.
- ❖ Balance.

4.2.3.2 Account Type:

The data about the types of accounts in the bank, such as:

- ❖ The account type number.
- ❖ The account type name.
- ❖ Some notes about the accounts.

4.2.4 Currency Data:

These are information about the currencies that are used in the bank, and their exchange prices, such as:

- ❖ Currency number.
- ❖ Currency name.
- ❖ Exchange price.

4.2.5 Transactions Data:

These are information about the transactions of money between the accounts in the bank done by customers, such as:

- Transaction number.
- Transaction name.
- Account number.
- Account type number.
- Curreny number.
- The date of the transaction.
- The amount of money used in the transaction.
- Other details about the transaction.

4.3 System Outcomes

The system outcomes define a set of reports that ease the the way of communication between the bank and its customers, that means that the customer can do and finish some of his work with the bank without going there. He can just enter the web page and finish his work also to take the information he needs throuh the following reports:

- A report that gives the balance of the account.
- A report that announce about transferring money from account to account, and the amount of money that has been transferred.

- A report that gives information about the currency exchange rates.
- A report that gives the account state within a certain period of time, showing all transactions that have been done with the date for each one.
- A report that announce about ordering a cheque book.
- Also the system will generate messages for errors that may happen while doing any activity in the web page, which helps the customer to correct himself.

4.4 Design Models

4.4.1 Entity Relationship (ER) Model

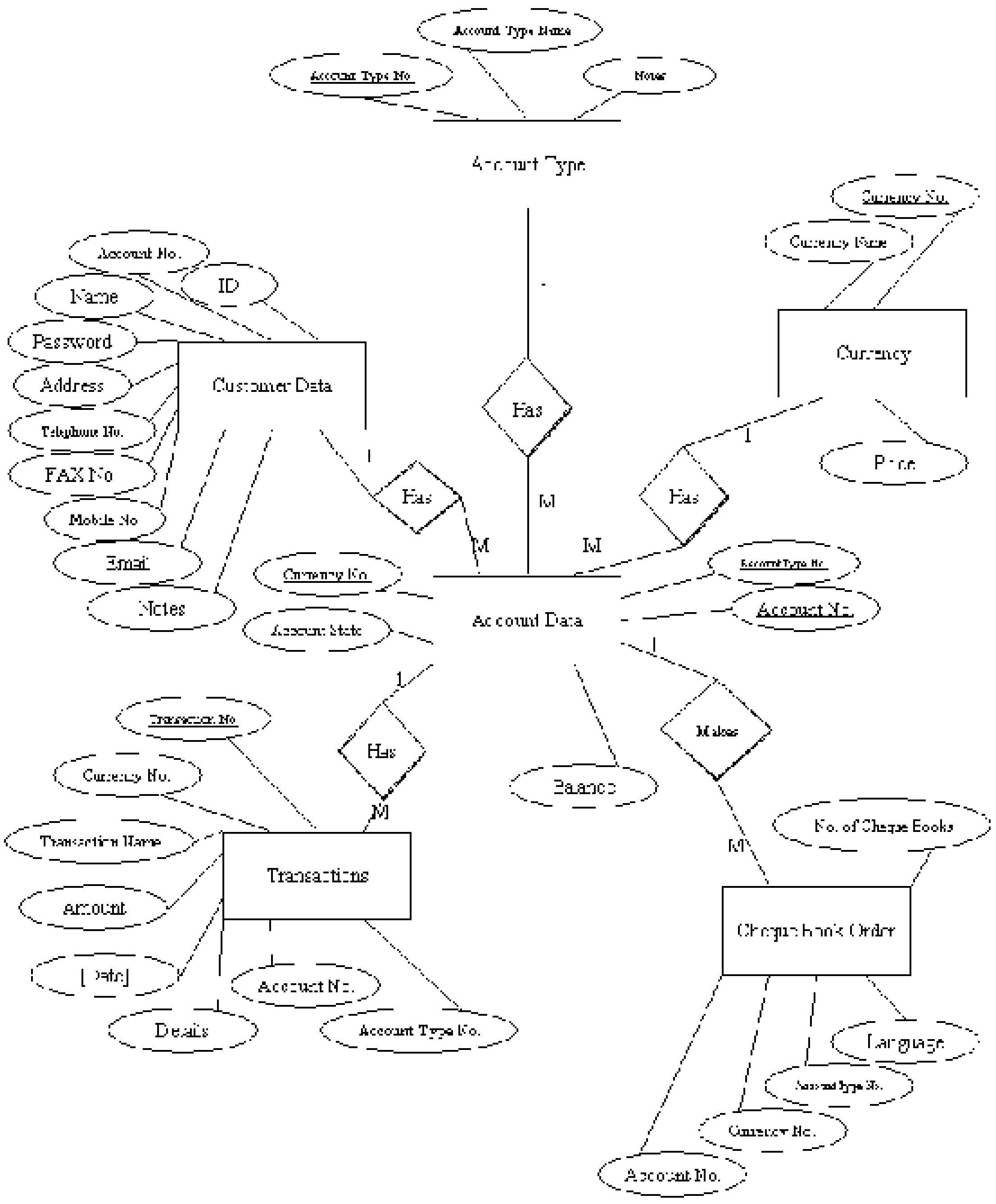


Figure 4.1: ER Model

4.4.2 Data Flow Diagrams

4.4.2.1 Entrance Process

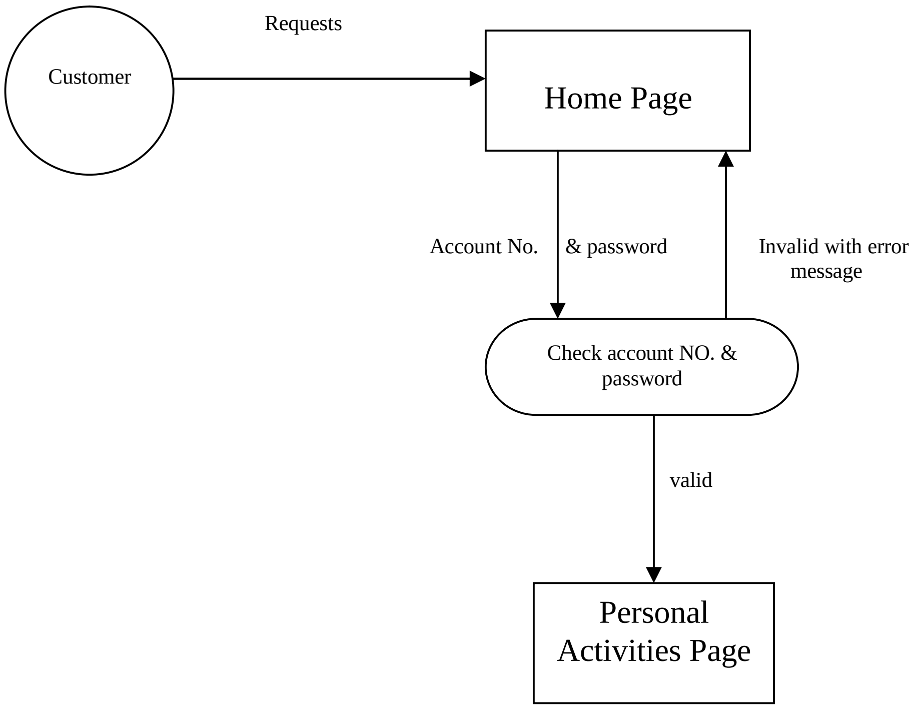


Figure 4.2: Entrance process

4.4.2.2 Check Balance Process

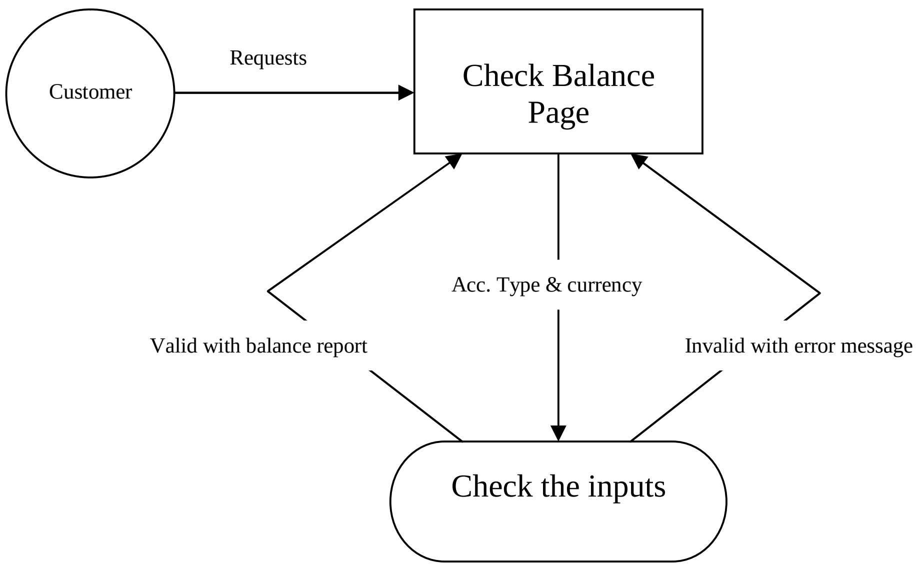


Figure 4.3: Check Balance Process

4.4.2.3 Transfer Money Process

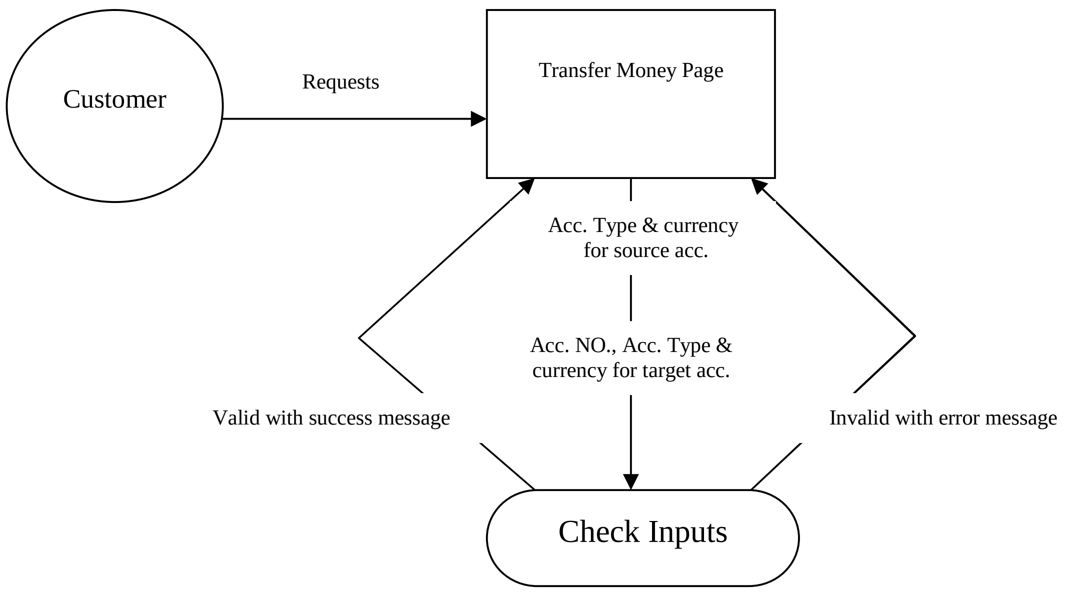

Customer
Requests
Transfer Money Page
Acc. Type & currency for source acc.
Acc. NO., Acc. Type & currency for target acc.
Valid with success message
Invalid with error message
Check Inputs

Figure 4.4: Transfer money process

4.4.2.4 Currency Prices Process

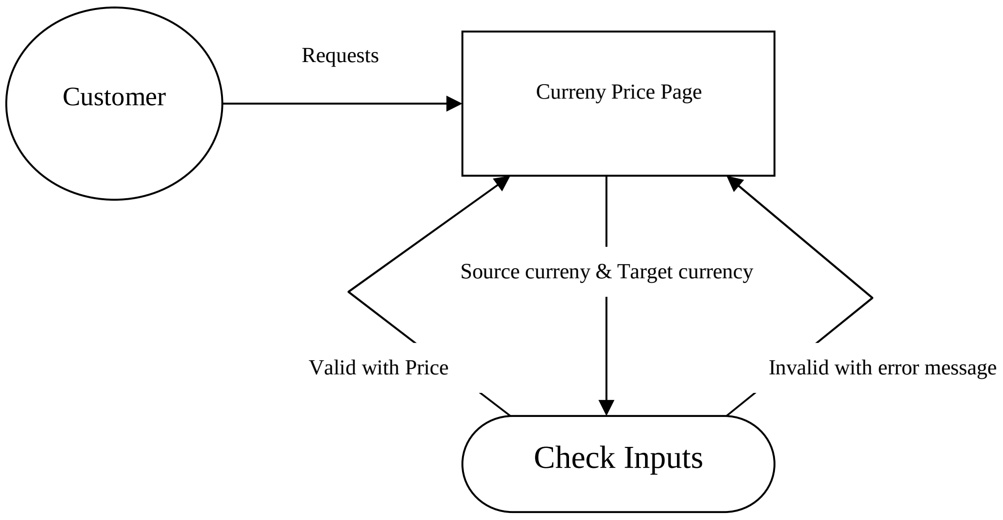


Figure 4.5: Currency Price Process

4.4.2.5 Account State Process

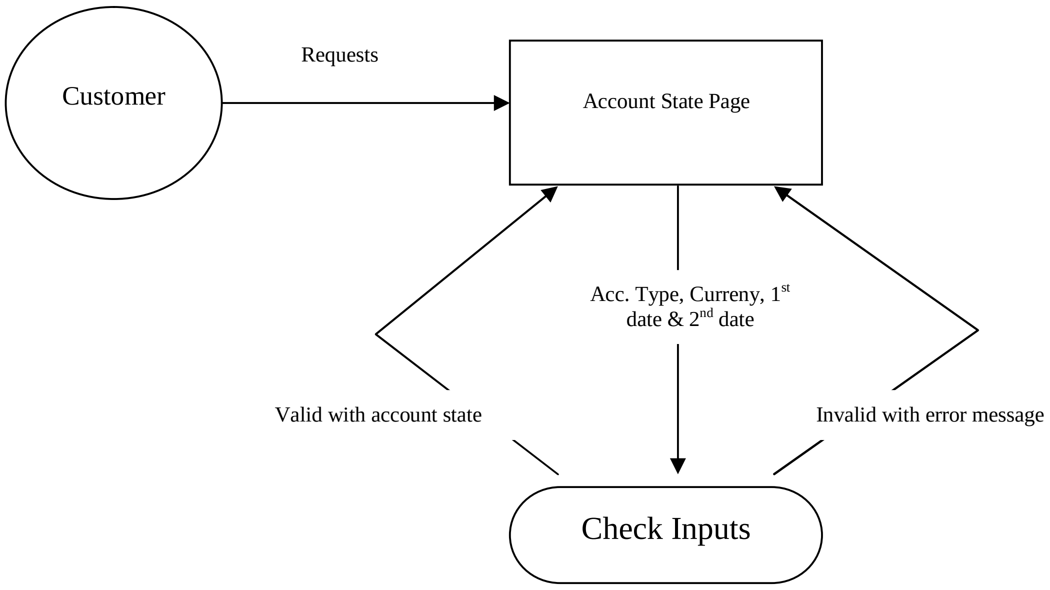


Figure 4.6: Account State Process

4.4.2.6 Cheque Book Order Process

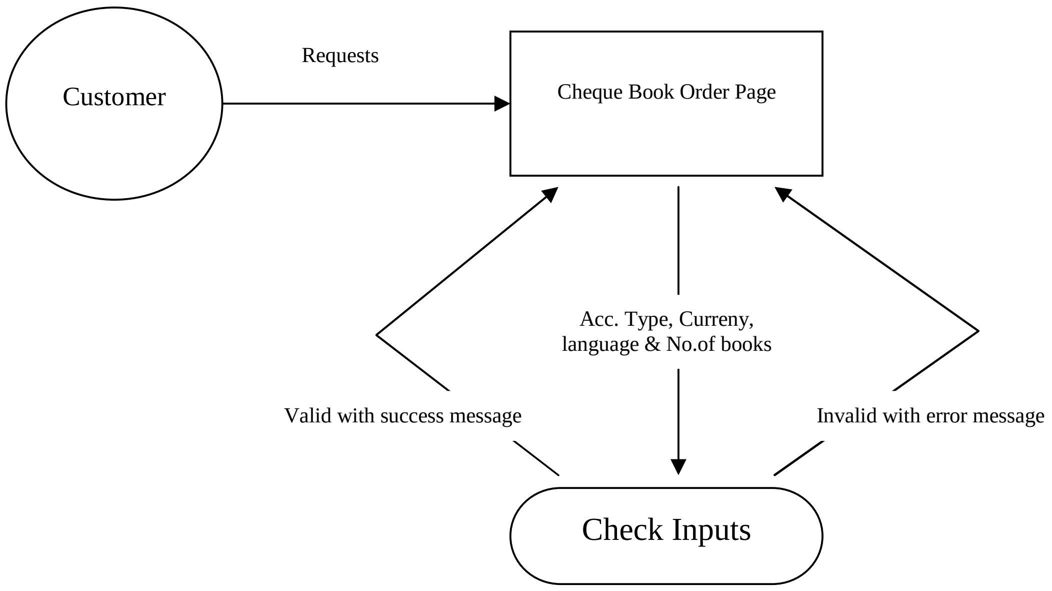

Customer
Requests
Cheque Book Order Page
Acc. Type, Curreny, language & No.of books
Valid with success message
Invalid with error message
Check Inputs

Figure 4.7: Check Book Order Process

4.4.4 System Flowchart

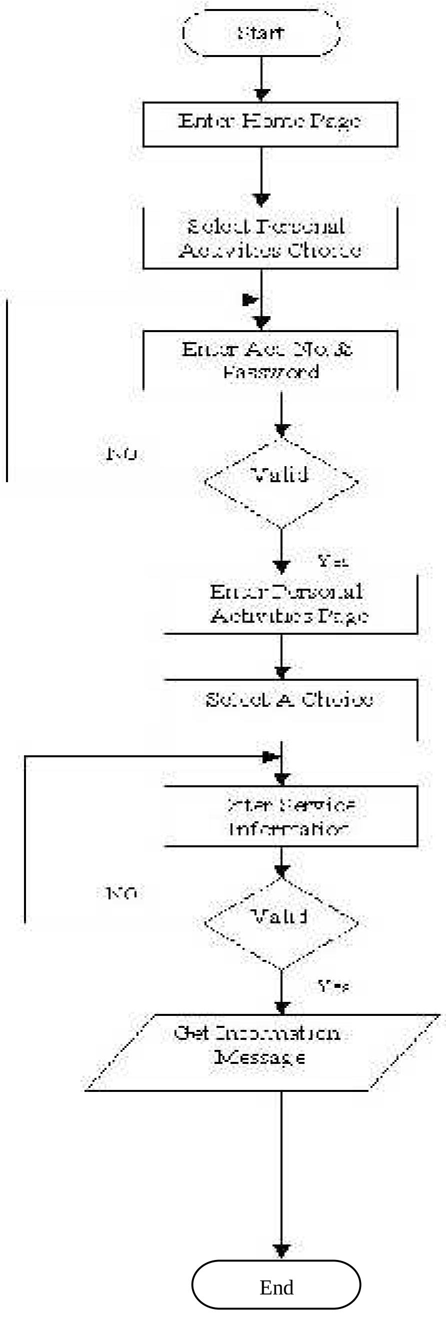


Figure 4.8: System Flowchart

4.4.5 System Interface

This section shows the system user interfaces used in our project. The figures shown represent an imagination for the system screen. The real system screens will be a little bit different.

4.4.5.1 The Default Web Page

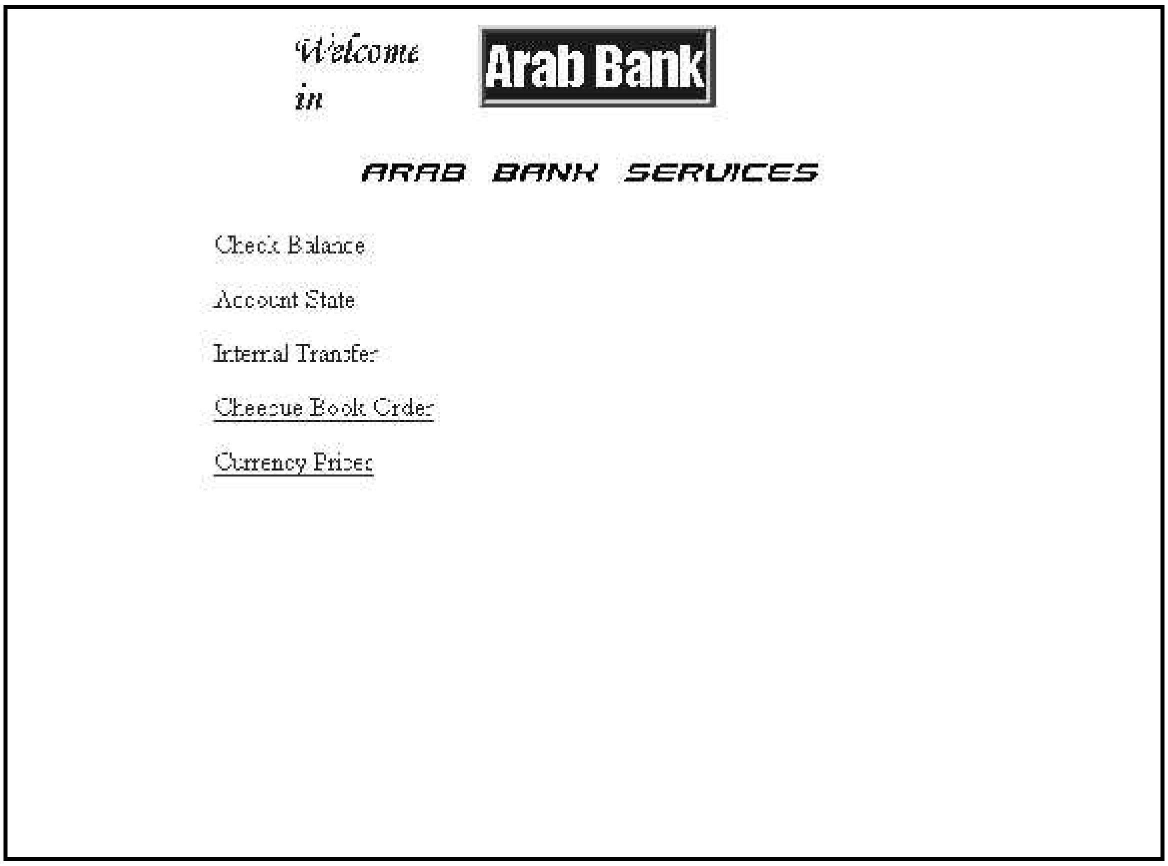


Figure 4.9: The Default Web Page

4.4.5.2 Login Page

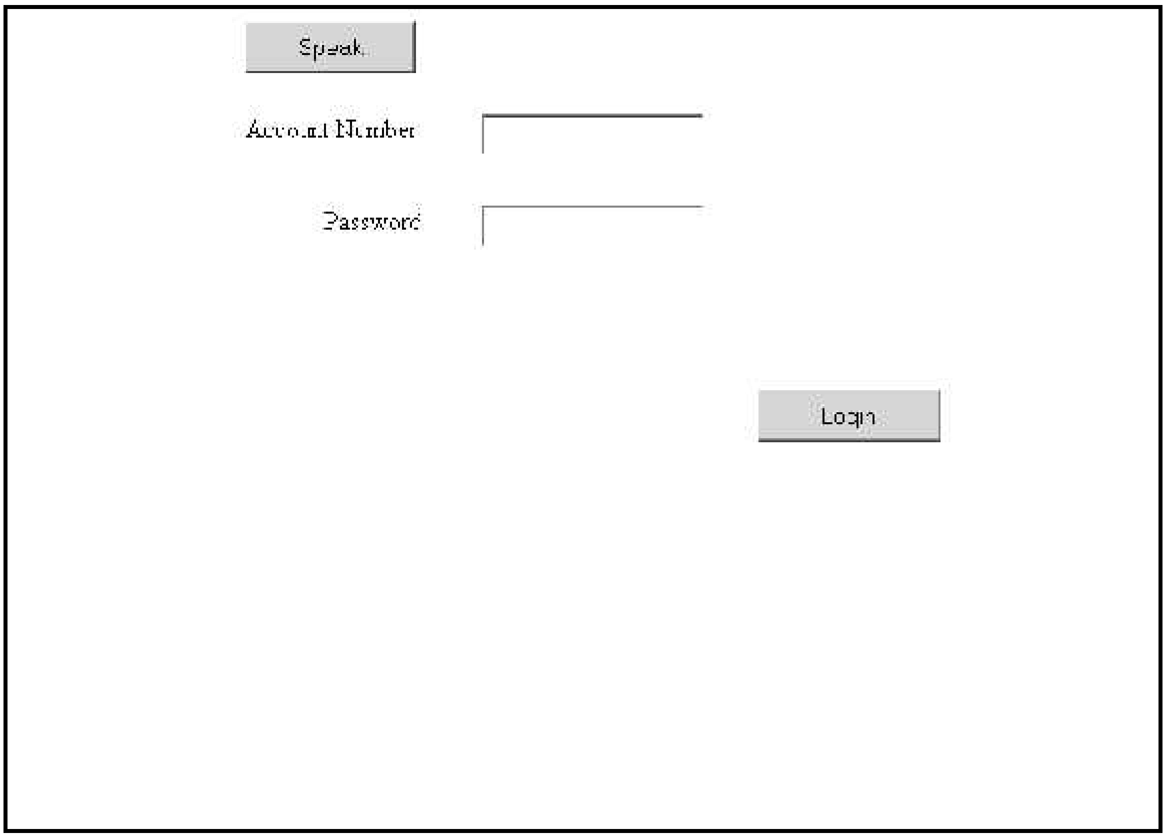


Figure 4.10: Login Page

4.4.5.3 Check Balance Page

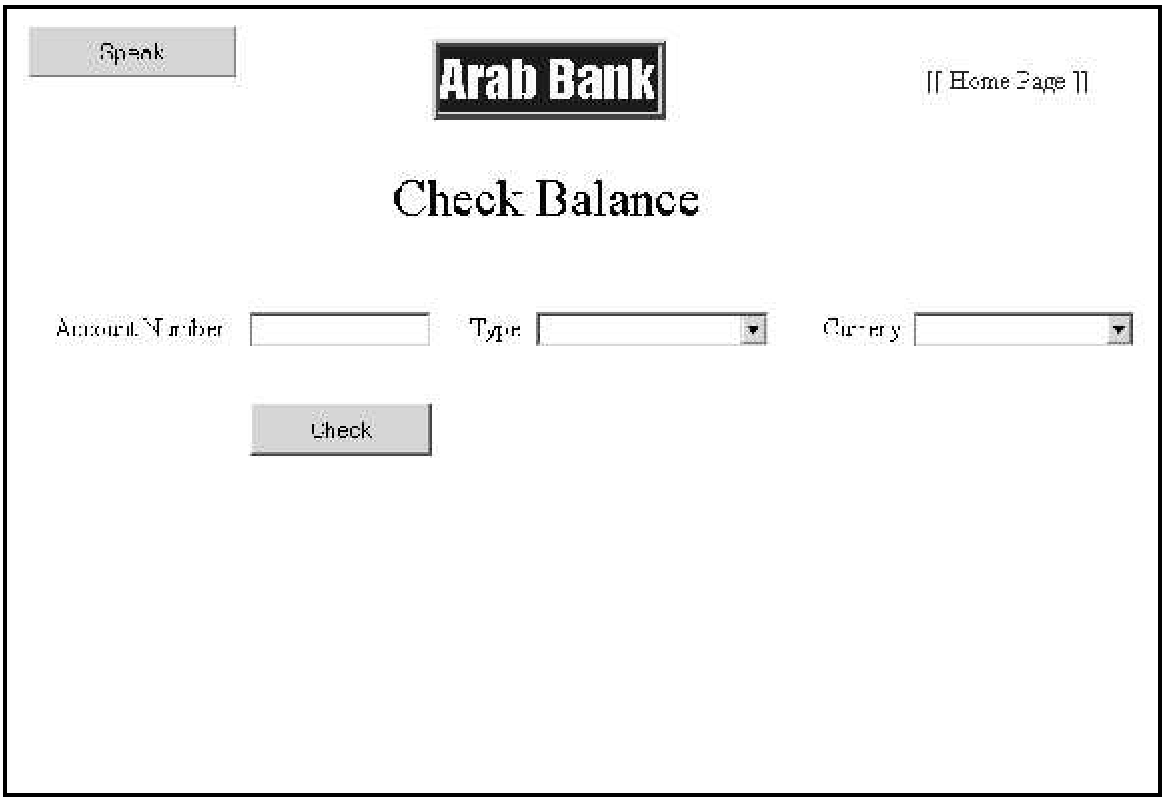
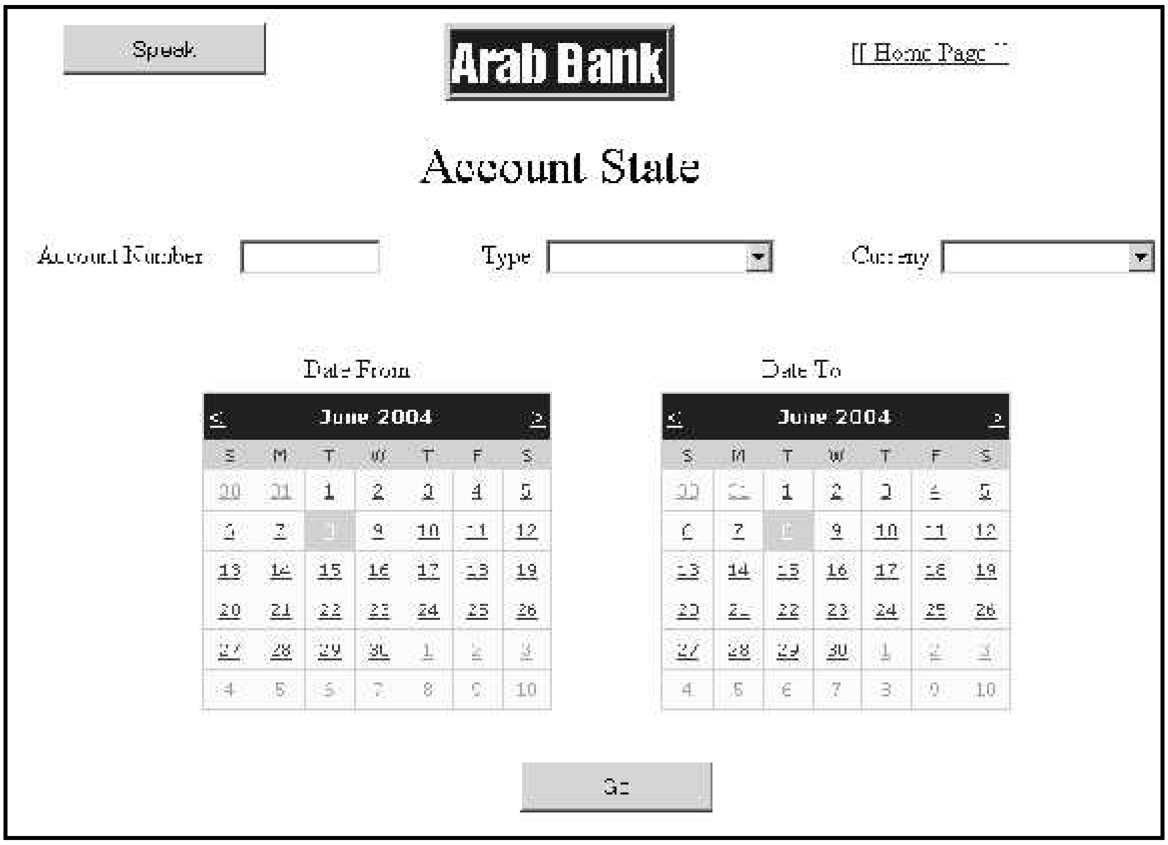


Figure 4.11: Check Balance Page

4.4.5.4 Account State Page



<			June 2004			>
S	M	T	W	T	F	S
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3
4	5	6	7	8	9	10

<			June 2004			>
S	M	T	W	T	F	S
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3
4	5	6	7	8	9	10

Figure 4.12: Account State Page

4.4.5.5 Internal Transfer Page

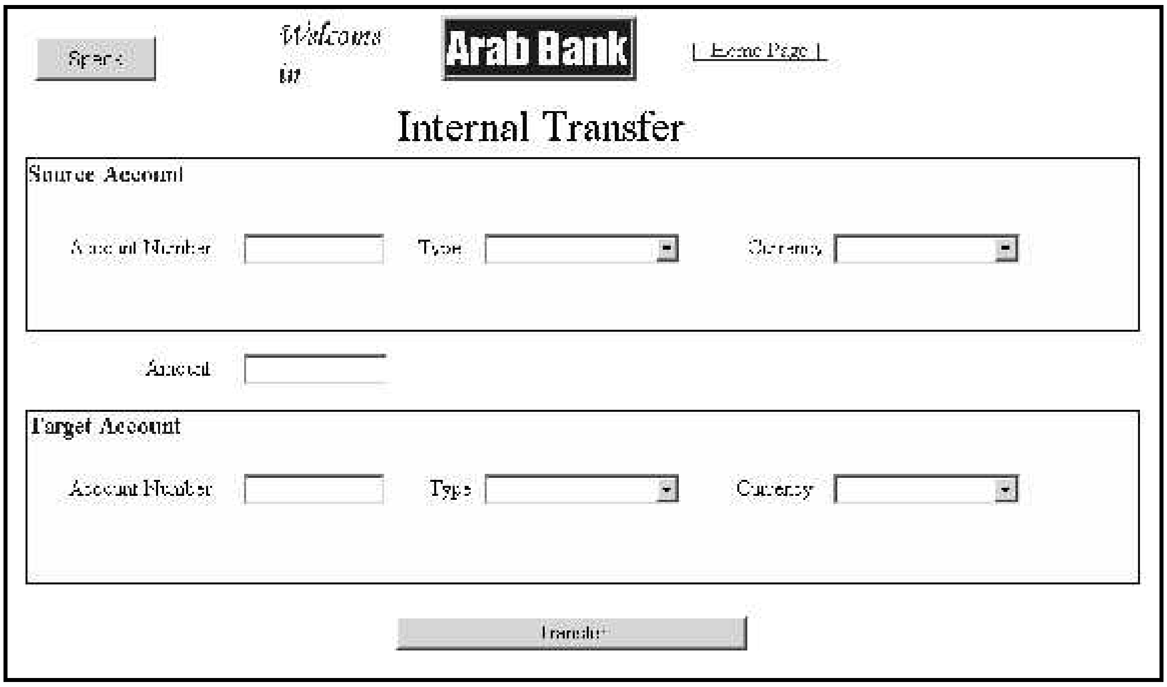
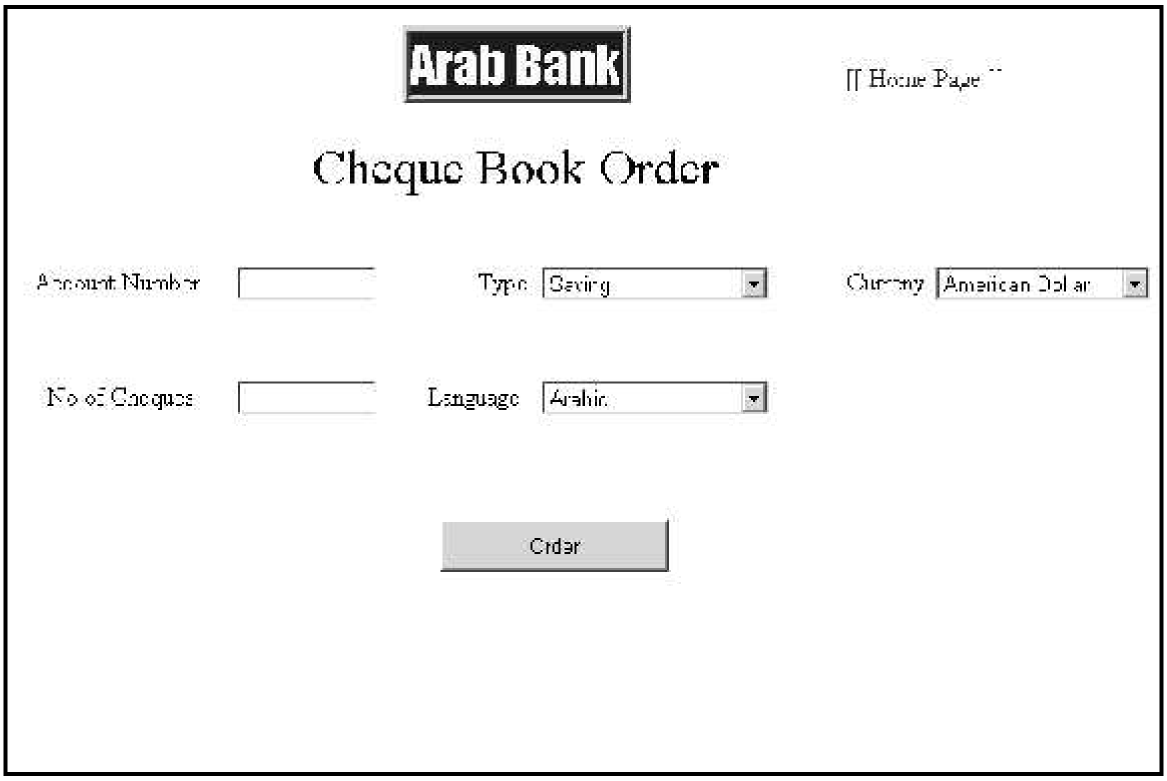


Figure 4.13: Internal Transfer Page

4.4.5.6 Cheque Book Order Page



Cheque Book Order

Account Number		Type	Saving	Currency	American Dollar
No of Cheques		Language	Arabic		

Figure 4.14: Cheque Book Order Page

4.4.5.7 Currency Prices Page

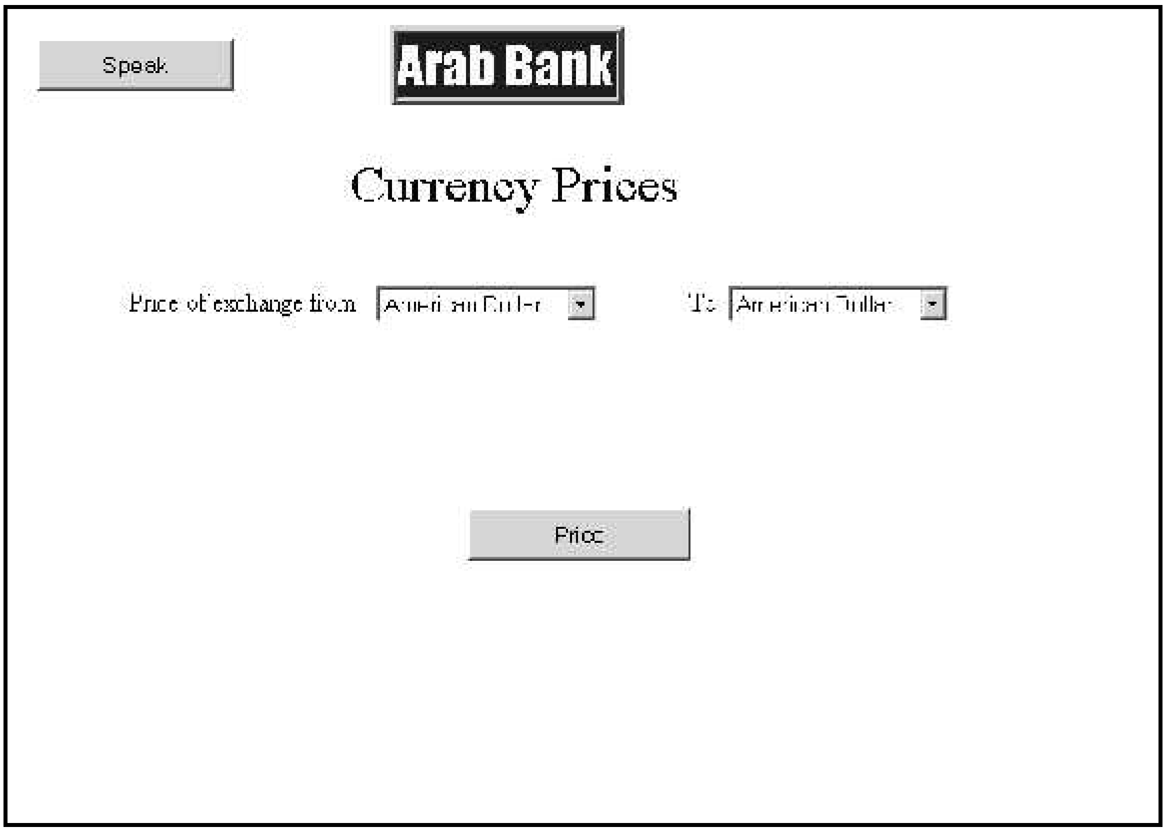


Figure 4.15: Currency Prices Page

Chapter Five

Implementation

Chapter Five
Implementation

5.1 Introduction

The implementation of the system needs some knowledge of Bank Services and treatments. It also needs the developer to have a good experience in Visual Studio.NET and Database management systems.

The implementation of the system involves the following activities:

1. Install Windows XP Professional with Service Pack 1
2. Install the following services:
   - Internet Information Server IIS 6.0
   - Front Page Server Extension 2002
   - World Wide Web Publishing Service
   - HTTP SSL
3. Install Visual Studio.NET 2003 with .NET Framework 1.1
4. Install Speech Application SDK Beta 3 Version 1.0

5.2 Database Design

After the installation of the required software on the server, we began to design the database of the system. We began with a flat database that consists of all fields and attributes required for the system's data. The flat database is normalized so that the database tables are in the third normal form.

5.2.1 Database Tables

Table 5.1 shows the data dictionary that contains the names of the tables in the database and there descriptions.

Table 5.1: Database Tables

Table Name	Description
CustomerData	This table contains personal information about the customer such as name, address … etc.
AccountData	This table contains information about the account of the customer. This includes available balance.
AccountType	This table holds the different types of accounts
Currency	This table holds the different currencies used in the Bank.
Transactions	This table contains a record of the transactions for all customers
ChequeBookOrder	This table contains the cheque book orders done by customers

Table 5.2: CustomerData Table

Field Name	Description	Type	Length	PK	FK	Related Tables	Constraints
Name	Customer Name	Nvarchar	50	No	No		Not Null
Password	Login password	Nvarchar	50	No	No		Not Null
ID	ID Number	Nvarchar	50	No	No		Not Null
AccountNo	Account number	Nvarchar	50	Yes	No		Not Null
Address	Customer address	Nvarchar	50	No	No		
TelephonNo	Telephone number	Int	10	No	No		
MobileNo	Mobile number	Int	10	No	No		
FaxNo	Fax number	Int	10	No	No		
Email	Email address	nvarchar	50	No	No		
Notes	Extra Notes	Nvarchar	50	No	No		

Table 5.3: AccountData Table

Field Name	Description	Type	Length	PK	FK	Related Tables	Constraints
AccountType No	Account Type Number	Int	2	Yes	Yes	AccountType	Not Null
Account No.	Account Number	Nvarchar	50	Yes	Yes	CustomerData	Not Null
Currency No.	Currency	Int	2	Yes	Yes	Currency	Not Null

	Number						
Account State	Account State	Nvarchar	50	No	No		Not Null
Balance	Holds Balance	Numeric	9	No	No		Not Null

Table 5.4: AccountType Table

Field Name	Description	Type	Length	PK	FK	Related Tables	Constraints
AccountTypNo	Account Type Number	Int	2	Yes	No		Not Null
AccountType Name	Account Type Name	Nvarchar	50	No	No		Not Null
Notes	Notes	Nvarchar	50	No	No		

Table 5.5: Currency Table

Field Name	Description	Type	Length	PK	FK	Related Tables	Constraints
Currency No.	Currency Number	Int	2	Yes	No		Not Null
Currency Name	Currency Name	Nvarchar	50	No	No		Not Null
Price	Exchange Price	Real	4	No	No		Not Null

Table5.6: Transactions Table

Field Name	Description	Type	Length	PK	FK	Related Tables	Constraints
Transaction No	Transaction Number	Numeric	5	Yes	No		Not Null
Transaction Name	Transaction Name	Nvarchar	50	No	No		Not Null
Currency No.	Currency Number	Int	2	No	Yes	Currency	Not Null
Account No.	Account Number	Nvarchar	50	No	Yes	Customer Data	Not Null
Account Type No.	Account Type Number	Int	2	No	Yes	Account Type	Not Null
Amount	Amount of Money Transferred	Numeric	9	No	No		Not Null
TDate	Date of Transaction	Datetime	8	No	No		Not Null
Details	More Information	Nvarchar	50	No	No		Not Null

Table 5.7: Cheque Book Order Table

Field Name	Description	Type	Length	PK	FK	Related Tables	Constraints
OrderNo.	Order Number	Int	4	Yes	No		Not Null
Account No.	Account Number	Nvarchar	50	No	Yes	Customer Data	Not Null

Currency No.	Currency Number	Int	2	No	Yes	Currency	Not Null
Account Type No.	Account Type Number	Int	2	No	Yes	Account Data	Not Null
Language	The Language of The Cheque Book	Nvarchar	50	No	No		Not Null
Quantity	Number of Cheque Books	Int	2	No	No		Not Null

5.2.2 Database relationships

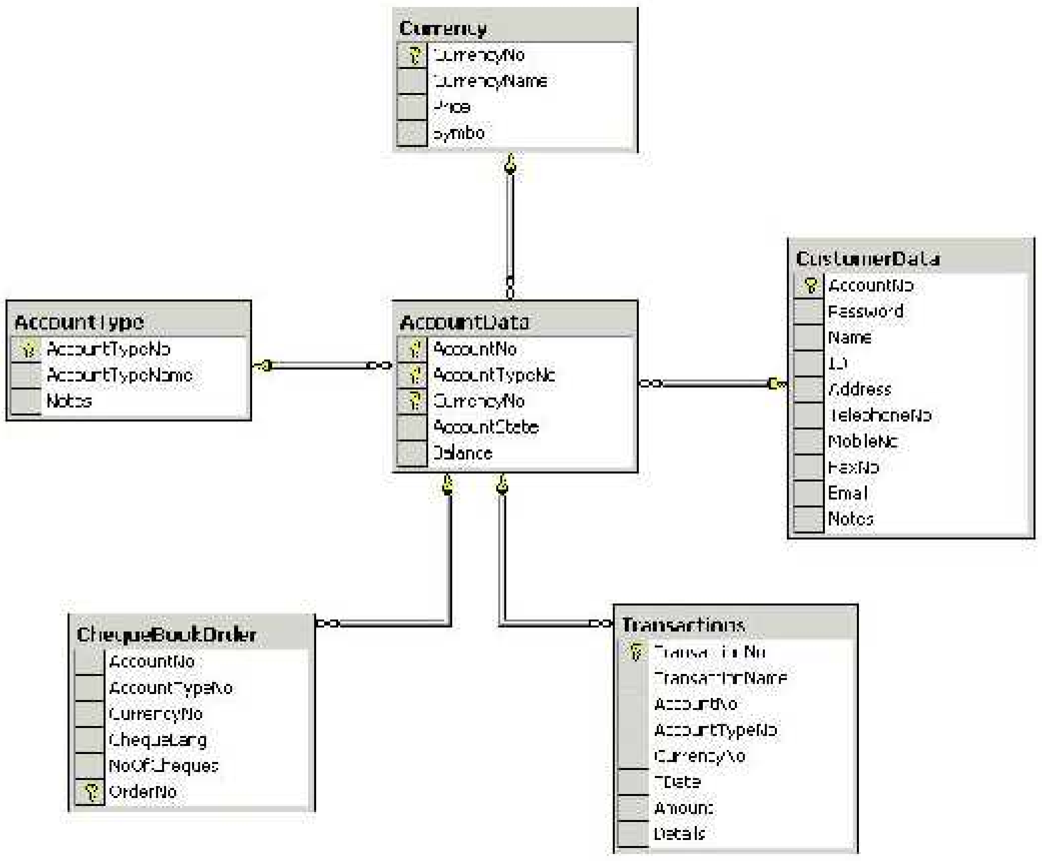


Figure 5.1: Database relationships

5.2.3 SQL Server 2000 Stored Procedures

A very efficient tool provided by SQL server 2000 is the use of stored procedures. Using this tool has many advantages such as:

- Modular Programming: the procedure is written once and stored on the database server. After that it can be called any number of times.
- Increase Database Security: using stored procedures provides increased security for database by limiting direct access.
- Faster Execution: Using stored procedures can be faster than direct database access if a procedure requires a large amount of Transact-SQL code.
- Reduce network traffic: sending the calling statement over the network rather than sending the whole code reduces network traffic.

Table 5.8 shows the stored procedures used in our system.

Table 5.8: The stored procedures used in the system

Stored Procedure Name	Description
CheckBalance	Check the available balance for a specific Account
GetState	Get the account state between two different dates
Transfer	Make internal transfer of money from one account to another
OrderChequeBook	Register a cheque book order done by customer on the database

GetCurrencyPrice	Get the rate of exchange of a currency
CheckLogin	Check weather the customer’s account number and password are valid.

5.3 Grammar Files Design

In order to make the web application speech enabled, a grammar must be added to each speech control. The Microsoft Speech Application SDK Version 1.0 Beta 3 (SASDK) provides Speech Grammar Editor to help simplify and expedite the process of creating grammars. Speech Grammar Editor uses connected, graphical elements to represent the grammar XML markup that defines grammar rules. Although grammar authors can create grammars manually using a plain text editor such as Notepad, this approach can become tedious, complex, and more prone to error when creating long grammar files or sets of grammar files. The graphical approach implemented by Speech Grammar Editor enables grammar authors to concentrate on creating the actual grammar rules instead of laborious tasks such as verifying proper element syntax.

Speech Grammar Editor is tightly integrated with Visual Studio .NET 2003, and runs completely within the Visual Studio .NET 2003 development environment. For more information about grammars and rules you can reference the Appendix.

The following subsections explains the Grammar Files used in the system.

5.3.1 Main Grammar

This grammar file (or grammar) contains some rules referenced frequently by other grammars. It can be thought of as custom library. It contains the following rules:

5.3.1.1 Type Rule

This rule is designed to recognize the Account type input by user. The user can speak to the computer and say one choice of the different account types such as saving, current … etc. Figure 5.2 shows the type rule.

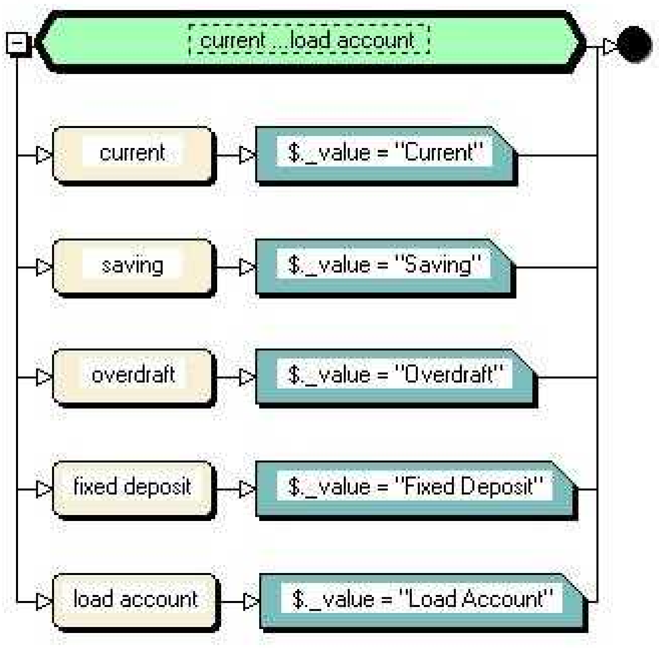


Figure 5.2 : Account Type rule

5.3.1.2 Currency Rule

This rule is used to recognize the currency chosen by user. It’s shown below in Figure 5.3 . As shown user can choice one of the three currencies.

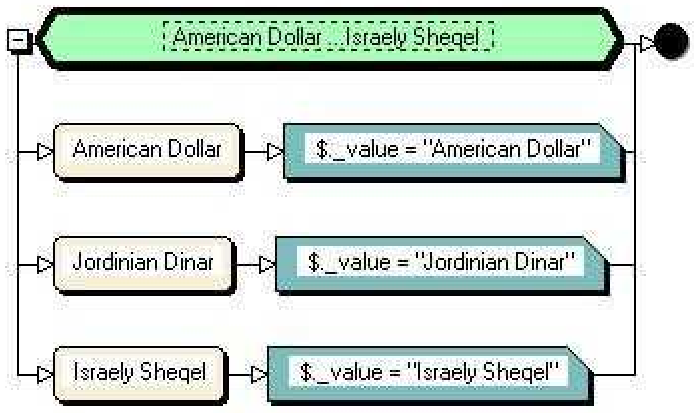

American Dollar ...Israely Sheqel
American Dollar
$._value = "American Dollar"
Jordinian Dinar
$._value = "Jordinian Dinar"
Israely Sheqel
$._value = "Israely Sheqel"

Figure 5.3 : Currency rule

5.3.2 Login Grammar

This grammar recognizes the account number and password spoken by the customer when he wants to login to the system. Figure 5.4 shows the Login Rule. As shown to recognize the account number a rule called Digit6 is called from the Grammar Library. It is designed to recognize a number consists of six digits. The password must be at least 6 digits and at most 8 digits. So, one of three rules may be called from Grammar Library: Digit6, Digit7, and Digit8.

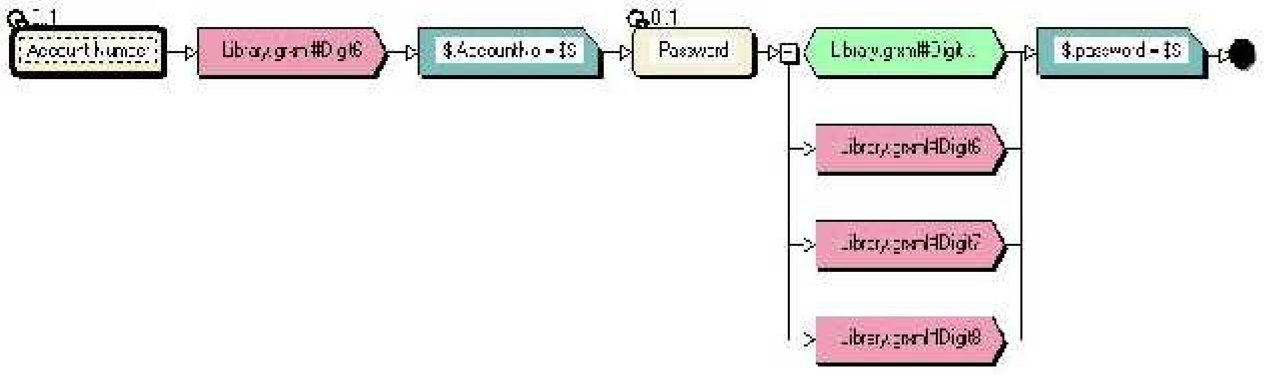


Figure 5.4: Login Grammar

5.3.3 Check Balance Grammar

This grammar is used to recognize voice input by user when he wants to check his balance. As mentioned before, when the user wants to check his balance, he needs to speak the Account type and currency. Of course, the Account Number is already entered in Login page. This grammar contains only one rule shown in Figure 5.5.

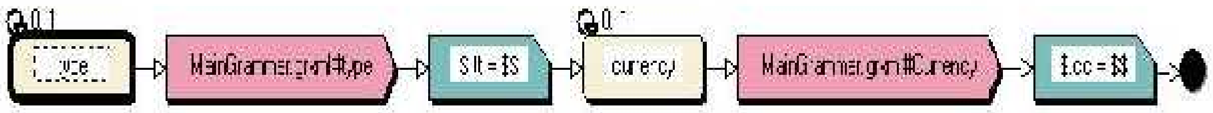


Figure 5.5: Check Balance Rule

As shown in the figure, the phrase elements (type & currency) may be spoken by user or may not i.e. user can say like this “type current currency American Dollar”. Or he can simply say “current American Dollar”. This makes the system more flexible and can understand different patterns of user inputs.

The rule references shown in the same figure refer to (or call) the Type and Currency rules found in the Main Grammar.

5.3.4 Account State Grammar

This grammar is used to recognize voice input by user when he wants to get an Account State report. Here, the user needs to speak the account type, currency, 1st date (Date From) and 2nd date (Date To). Figure 5.6 shows the rule used to do so.

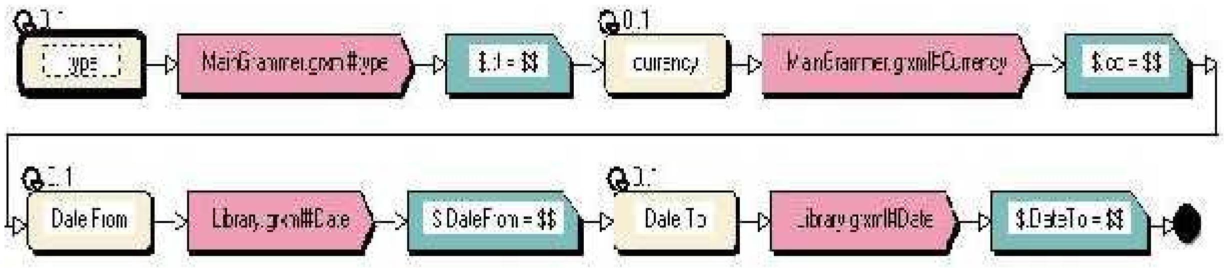


Figure 5.6 Account State Rule

For more flexibility the user may or may not speak the phrases shown in the rule.

5.3.5 Internal Transfer Grammar

This grammar is used to recognize voice input by user when he wants to make an internal transfer. The user needs to speak the account type and currency for both the source and the target accounts in addition to account number for the target account. He also speaks the amount of money he wants to transfer. Figure 5.7 shows the Internal Transfer Rule.

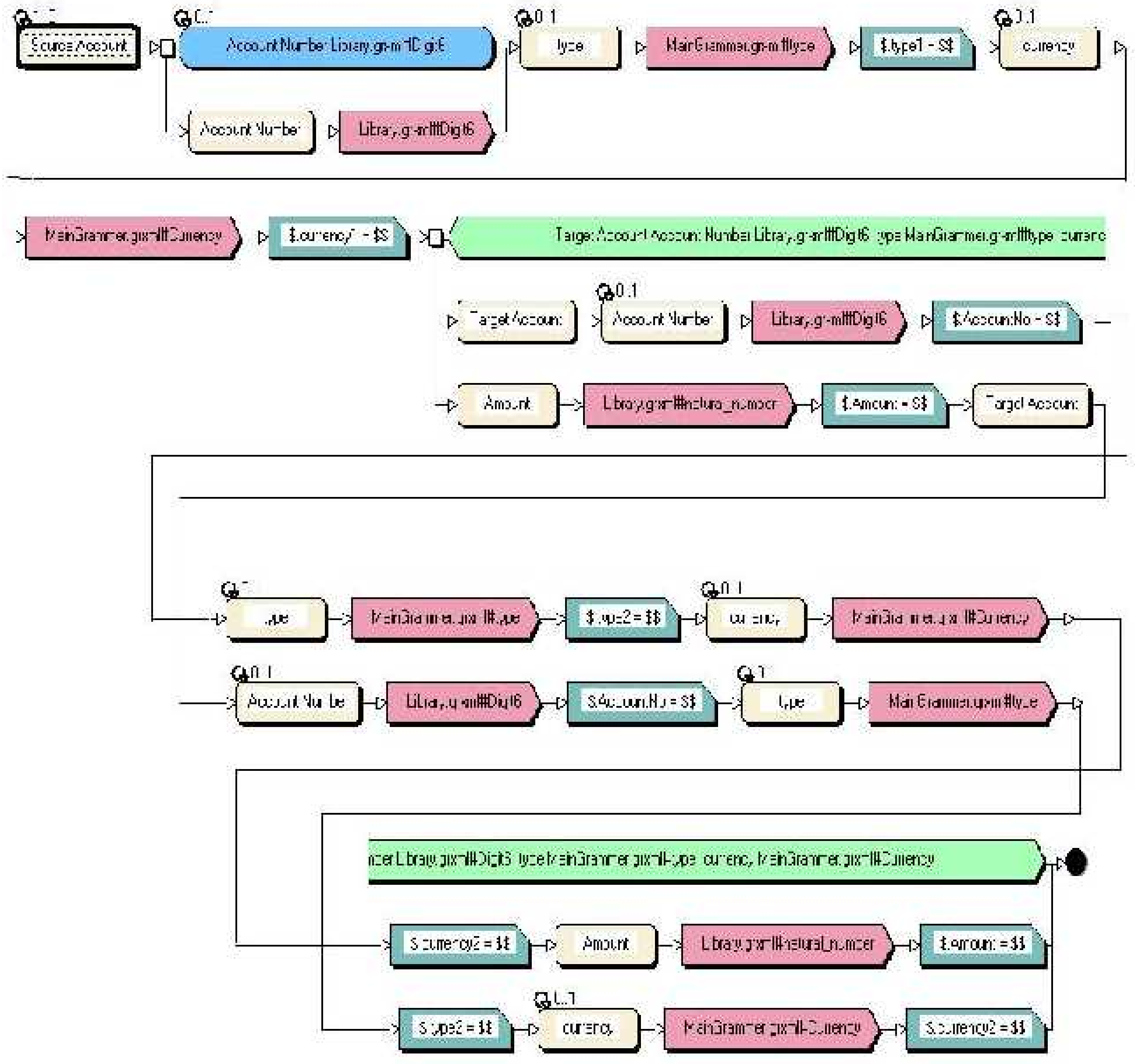
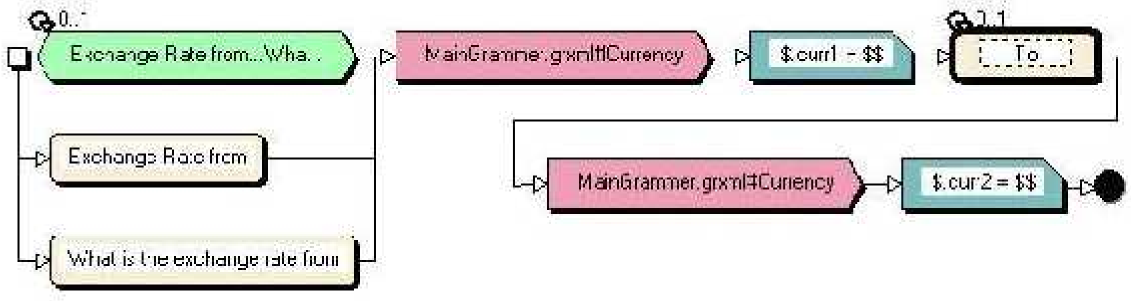


Figure 5.7 : Internal Transfer Rule

5.3.6 Exchange Rate Grammar

This grammar recognizes two currency names spoken by user and gives results to the system to calculate the exchange rate. Figure 5.8 shows the Exchange Rule. In this rule the user can ask the system like this: “ What is the exchange rate from American Dollar to Israeli Sheqel”. Or he can say “ price of exchange from American Dollar to Israeli Shekel”. He also may simply say “ American Dollar to Israeli Sheqel”.



0..*
Exchange Rate from...Wha...
Exchange Rate from
What is the exchange rate from
MainGrammer.grxml#Currency
$.curr1 = $$
0..1
To
MainGrammer.grxml#Currency
$.cur2 = $$

Figure 5.8: Exchange Rate Rule

5.3.7 Cheque Book Order Grammar

This grammar recognizes speech input by the user when he wants to order a cheque book. To do so, the user needs to speak the account type, currency, number of cheques, and the language of cheque. Figure 5.9 shows the rule used for that.

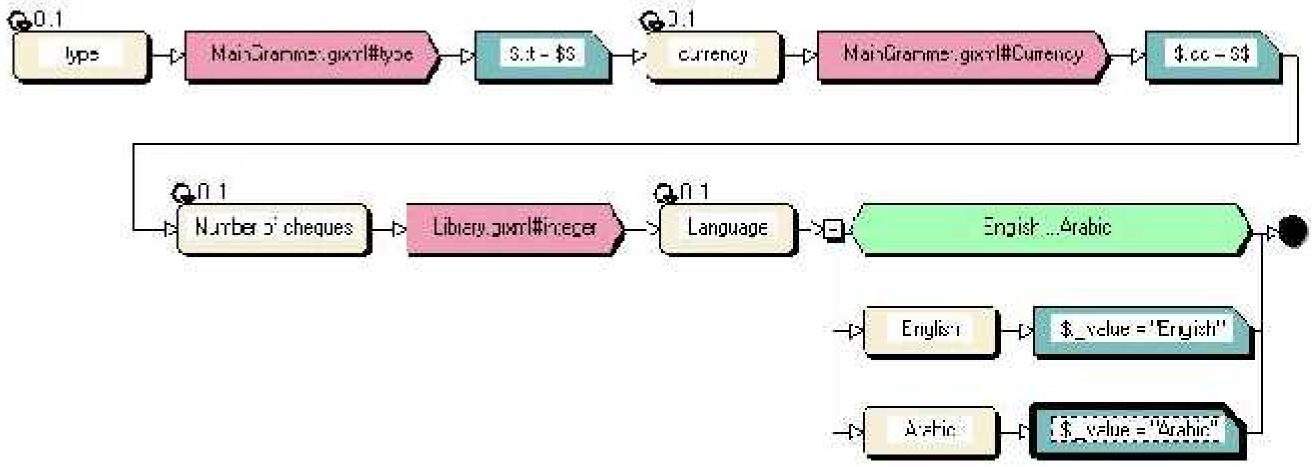


Figure 5.9: Check Book Order Rule

5.4 Implementation Steps

In order to implement the project, a number of steps must be followed.

First of all, the computer is partitioned using NTFS file system, which sustains folder security and enhances file encryption. Windows XP professional with SP 1 is installed. This operating system has a higher compatibility and consistency with the .NET Framework.

Visual Studio.NET is our development environment. It presents new window types, new way to manage those windows, and new integration with Internet content.

Before installing Visual Studio .NET, some components must be installed before. The following components are required to guarantee a correct operation for Visual Studio .NET environment:

1. Microsoft .NET Framework
2. Internet Information Service IIS 6.0
3. Front Page 2002 Server Extensions
4. Internet Explorer with Add-in tools

After that, Visual Studio .NET Enterprise Edition and Microsoft SQL Server 2000 Enterprise Edition are installed. The SQL Server must include SQL Enterprise Manager and SQL Query Analyzer.

The database is build using Microsoft SQL Server 2000. Tables are built to be immediately in the 3rd Normal level.

Now, the grammars needed in the system are built using Grammar Editor. Next, web forms are built one by one and Speech Controls are added to each web form and configured as necessary to make the web pages voice-enabled.

5.5 Implementing System Security

In our system, security is the most crucial issue since we are talking about a bank database access. Non authenticated access to this database will cause data stored about customers' accounts to be in danger.

Three phases of security are applied to the system to make sure that no one can access the database except authorized users.

5.5.1 SQL Server Stored procedures

The system uses many of SQL Server Stored Procedures. Using Stored Procedures provides more security for the database by limiting direct access. Only the tested and proven stored procedures that are developed by the owner of the database can access the database directly. Thus, there is a minimum risk of accidental damage to the structure or to the content of the database.

On the other hand, using SQL or Transact-SQL statements directly in the ASP.NET code is a security risk because such statements can give a hacker some information about the database and its structure.

5.5.2 Forms-Based Authentication

ASP.NET implements authentication through authentication methods. ASP.NET authentication methods contain the code that is necessary to authenticate the user’s credentials.

ASP.NET supports three types of authentication methods:

- Windows based authentication
- Forms-based authentication
- Microsoft Passport authentication

The Web Database Access System implements the second technique; Forms-based authentication.

Forms-based authentication refers to a system where non-authenticated requests are redirected to a Web page that contains a form by using Hyper Text Transfer Protocol (HTTP) client-side redirection. The user provides credentials and submits the form. If the application validates the credentials on the form, the system issues an authentication cookie to the user. Subsequent requests from the user are issued with the authentication cookie in the request headers, and then the user is authenticated based on those request headers.

Chapter six
System Testing

6.1 Unit Testing

6.2 Sub-System Testing

6.3 System Testing

6.4 Acceptance Testing

Chapter six
System Testing

The system must be tested to ensure that it will operate correctly. It may be tested immediately after finishing design and implementation or during implementation. This chapter illustrates the testing process of the system.

6.1 Unit Testing

Since the system is divided into units that are independent from each other, the units could be tested one by one. Testing one unit does not affect the others.

First the database is checked using SQL Server Query Analyzer. The stored procedures are tested by assigning values to the input parameters and checking results. This test was applied to all database items and gave the desired results.

After a connection to the Bank data base is established, it is tested to make sure it is operating correctly.

The Speech Application SDK provides a Grammar Editor. This tool is used to build grammars and rules and provides a mechanism to test weather rules are working properly or not. All rules in the grammars were checked and gave the required results.

6.2 Sub-System Testing

Each sub-system needs to be tested individually so as to give the desired output. The sub-systems are services provided by the Bank such as checking balance, checking account state … etc. Each service has a web page shown in System Interface in Chapter Four. These sub-systems are tested by inputting data through the keyboard and mouse (visually) and then tested by voice.

6.2.1 Login Sub-System

In the login web page, we input the Account Number and Password and for an authenticated user stored in the Bank database and check if it passes the login page or not. After visual checking, the login page is checked by inputting data by voice.

6.2.2 Check Balance Sub-System

In the Check Balance web page The Account Number will be already input by customer in the login page so we only need to input the Account Type and Currency. These values are first input visually and then by voice. Inputs are input and then results are checked.

6.2.3 Account State Sub-System

In the Account State web page, the Account Type, Currency, 1st Date, and 2nd Date are input visually first and then by voice. Results are shown and verified.

6.2.4 Internal Transfer Sub-System

In the Internal Transfer web page, the Source Account data, Target Account data, and the Amount of money to be transferred are input visually and then by voice. Results are shown and verified.

6.2.5 Currency Prices Sub-System

In the Currency Prices web page, the Source Currency and the Target Currency are input visually and then by voice. Results are shown and verified.

6.2.6 The Cheque Book Order Sub-System

In the Cheque Book Order web page the Account Type, Currency, Language of Cheque, and Number of Cheques are input visually and then by voice. Results are shown and verified.

6.3 System Testing

Sub-Systems are integrated to constitute the whole system. Valid data are input to the system to make sure it will function properly. Also invalid data is input to the system to check the system’s response.

The system is tested by inputting data visually and by voice. A number of persons involved in testing the system by voice to check the response of the system when different voices are input.

6.4 Acceptance Testing

Acceptance Testing is used to ensure that the system design is consistence with the system requirements. This test is passed successfully and all requirements are satisfied.

Chapter Seven

Conclusions and Future Work

Chapter Seven
Conclusions and Future Work

7.1 Conclusions

After designing and implementing this project the project worked properly and the user could interact with the web pages through voice.

The followings are the main conclusions gained from this study:

- Communicating with computer by voice is a new technology that will be widely used in the near future.
- The Microsoft .NET Speech Application SDK allows web developers to create, debug, and deploy speech-enabled ASP.NET applications.
- The Speech Engine used to recognize speech is intelligent enough to recognize different patterns of speech.
- Interaction with web pages by voice gives better results when the user is asked to select an item from a collection; since the number of available options is limited. But when the user want to dictate the computer the recognition results will contain more errors.

- Security is a very important issue when accessing databases remotely especially if the database contains sensitive information such as a Bank database.

7.2 Future Work

This project is one of the first projects that use Speech Application SDK to develop a speech-enabled web application. This software kit is a very new component from Microsoft that has a great wealth inside. The following suggestions may be helpful for future work:

- Voice only (telephony) applications can be developed using SASDK if a telephony card is available.

- A speech web application can be designed for educational requirements where users need to speak to computer and listen to it.

References

Books

[1] Microsoft Corporation, Course 2310B, Developing Microsoft ASP.NET Web Applications Using Visual Studio .NET. Microsoft Press, USA, 2002.

[2] Microsoft Corporation Exam 70-229. Microsoft SQL Server 2000 Database Design and Implementation, Microsoft Press, USA, 2003.

Web Sites

[3] http://www.asp.net

[4] http://www.asptoday.com

[5] http://www.asp101.com

[6] http://www.15seconds.com

Documentation (Help)

[7] Microsoft Speech Application SDK 1.0 Beta 3 Help

[8] MSDN Library For Visual Studio .NET 2003

Appendices

Appendix A

System Code

A.1 ASP.NET Code

A.1.1 Web.Config File

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <system.web>

    <compilation defaultLanguage="vb" debug="true" />

    -->
    <customErrors mode="RemoteOnly" />


<authentication mode="Forms">
      <forms name=".namesuffix" loginUrl="wflogin.aspx" />
</authentication>



    <authorization>
<deny users="?" />
</authorization>


    <trace enabled="false" requestLimit="10" pageOutput="false"
traceMode="SortByTime" localOnly="true" />


    <sessionState
            mode="InProc"
            stateConnectionString="tcpip=127.0.0.1:42424"
            sqlConnectionString="data
source=127.0.0.1;Trusted_Connection=yes"
            cookieless="false"
            timeout="20"
    />

    <globalization requestEncoding="utf-8" responseEncoding="utf-8" />

  </system.web>

 <location path="wfBank.aspx">
<system.web>
<authorization>
<allow users="?" />
</authorization>
```

```
</system.web>
</location>

</configuration>
```

A.1.2 Code Behind of Login Page

```
Private Sub cmdLogin_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdLogin.Click
        Dim cmdLogin As New SqlCommand("select * from CustomerData",
SqlConnection1)
        SqlConnection1.Open()
        Dim dr As SqlDataReader
        dr = cmdLogin.ExecuteReader

        Do While dr.Read
            If dr("AccountNo") = TextBox1.Text And dr("Password") =
TextBox2.Text Then
                'Session("AccountNo") = TextBox1.Text
                dr.Close()
                SqlConnection1.Close()
                FormsAuthentication.Authenticate(TextBox1.Text,
TextBox2.Text)

FormsAuthentication.RedirectFromLoginPage(TextBox1.Text, False)
                Response.Redirect("wfHome.aspx")
            End If
        Loop


    End Sub
    Protected Sub login(ByVal o As Object, ByVal e As System.EventArgs)
        Dim num1, num2 As XmlNode
        num1 = Listen1.RecoResult.SelectSingleNode("AccountNo/text()")
        num2 = Listen1.RecoResult.SelectSingleNode("password/text()")

        TextBox1.Text = num1.Value
        TextBox2.Text = num2.Value

        Label3.Text = num2.Value

        Dim cmdLogin As New SqlCommand("select * from CustomerData",
SqlConnection1)
        SqlConnection1.Open()
        Dim dr As SqlDataReader
        dr = cmdLogin.ExecuteReader
        Do While dr.Read
            If dr("AccountNo") = TextBox1.Text And dr("Password") =
TextBox2.Text Then
                Session("AccountNo") = TextBox1.Text
                Response.Redirect(Session("page"))
            End If
        Loop

        dr.Close()
        SqlConnection1.Close()



    End Sub
```

A.1.3 Code Behind of Check balance Page

```
Imports System.Data
Imports System.Data.SqlClient
Imports System.Xml

Public Class frmBalance
    Inherits System.Web.UI.Page

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.SqlConnection1 = New System.Data.SqlClient.SqlConnection
        '
        'SqlConnection1
        '
        Me.SqlConnection1.ConnectionString = "workstation id=""SUNRISE-
ONAPWVQ"";packet size=4096;user id=sa;data source=""SUNRISE" & _
        "-ONAPWVQ"";persist security info=False;initial
catalog=Bank;password=center"

    End Sub
    Protected WithEvents SqlConnection1 As
System.Data.SqlClient.SqlConnection
    'Protected WithEvents DataSet11 As BankDataBase2.DataSet1
    Protected WithEvents TextBox1 As System.Web.UI.WebControls.TextBox
    Protected WithEvents Label2 As System.Web.UI.WebControls.Label
    Protected WithEvents Label3 As System.Web.UI.WebControls.Label
    Protected WithEvents Label4 As System.Web.UI.WebControls.Label
    Protected WithEvents Button1 As System.Web.UI.WebControls.Button
    Protected WithEvents ddlType As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents ddlCurrency As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents AccountListen As
Microsoft.Speech.Web.UI.Listen
    Protected WithEvents Label5 As System.Web.UI.WebControls.Label
    Protected WithEvents txtAccountNo As
System.Web.UI.WebControls.TextBox
    Protected WithEvents lbHome As System.Web.UI.WebControls.LinkButton
    Protected WithEvents DataGrid1 As
System.Web.UI.WebControls.DataGrid
    Protected WithEvents Label6 As System.Web.UI.WebControls.Label
    Protected WithEvents Label7 As System.Web.UI.WebControls.Label
    Protected WithEvents Image1 As System.Web.UI.WebControls.Image
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Image9 As System.Web.UI.WebControls.Image

    'NOTE: The following placeholder declaration is required by the Web
Form Designer.
    'Do not delete or move it.
    Private designerPlaceholderDeclaration As System.Object

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()
    End Sub

#End Region

    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Put user code to initialize the page here

        txtAccountNo.Text = Session("AccountNo")

        If (Not Page.IsPostBack) Then


            Dim cmdCurrency As New SqlCommand("select * from Currency",
SqlConnection1)
            Dim cmdType As New SqlCommand("select * from AccountType",
SqlConnection1)
            SqlConnection1.Open()

            Dim drCurrency As SqlDataReader
            drCurrency = cmdCurrency.ExecuteReader
            Do While drCurrency.Read
                ddlCurrency.Items.Add(drCurrency("CurrencyName"))
            Loop
            drCurrency.Close()

            Dim drType As SqlDataReader

            drType = cmdType.ExecuteReader

            Do While drType.Read
                ddlType.Items.Add(drType("AccountTypename"))
            Loop
            drType.Close()

            SqlConnection1.Close()
        End If

        'DataGrid1.DataBind()

    End Sub

    Protected Sub checkBalanceOnReco(ByVal o As Object, ByVal e As
System.EventArgs)

        ddlType.SelectedValue =
AccountListen.RecoResult.SelectSingleNode("tt/text()").Value
        ddlCurrency.SelectedValue =
AccountListen.RecoResult.SelectSingleNode("cc/text()").Value
        checkBal()

    End Sub


    Private Sub checkBal()
        Dim cmdBal As New SqlCommand
        Dim drBal As SqlDataReader

        cmdBal = New SqlCommand("CheckBalance", SqlConnection1)
        cmdBal.CommandType = CommandType.StoredProcedure
        cmdBal.Parameters.Add("@AccountNo", SqlDbType.NVarChar)
        cmdBal.Parameters.Add("@AccountTypeName", SqlDbType.NVarChar)
        cmdBal.Parameters.Add("@CurrencyName", SqlDbType.NVarChar)
        cmdBal.Parameters("@AccountNo").Direction =
ParameterDirection.Input
        cmdBal.Parameters("@AccountTypeName").Direction =
ParameterDirection.Input
        cmdBal.Parameters("@CurrencyName").Direction =
ParameterDirection.Input
```

```
        cmdBal.Parameters("@AccountNo").Value = txtAccountNo.Text
        cmdBal.Parameters("@AccountTypeName").Value =
ddlType.SelectedItem.Text
        cmdBal.Parameters("@CurrencyName").Value =
ddlCurrency.SelectedItem.Text

        SqlConnection1.Open()

        drBal = cmdBal.ExecuteReader()
        If Not drBal.HasRows Then
            Label6.Text = "Error! You dont have an acount with number "
& txtAccountNo.Text & " and Type:" & ddlType.SelectedItem.Text & " in "
& ddlCurrency.SelectedItem.Text

        End If

        DataGrid1.DataSource = drBal
        DataGrid1.DataBind()

        SqlConnection1.Close()

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        checkBal()
    End Sub

    Private Sub lbHome_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles lbHome.Click
        Response.Redirect("wfHome.aspx")
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Label6.Text = "button is clicked"
    End Sub
End Class
```

A.1.4 Code Behind of Account State

```
Imports System.Data
Imports System.Data.SqlClient
Imports System.Xml
```

```
Public Class wfAccountState
    Inherits System.Web.UI.Page

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.SqlConnection1 = New System.Data.SqlClient.SqlConnection
        Me.SqlCommand1 = New System.Data.SqlClient.SqlCommand
        '
        'SqlConnection1
        '
        Me.SqlConnection1.ConnectionString = "workstation id=""SUNRISE-
ONAPWVQ"";packet size=4096;user id=sa;data source=""SUNRISE" & _
        "-ONAPWVQ"";persist security info=False;initial
catalog=Bank;password=center"
        '
        'SqlCommand1
        '
        Me.SqlCommand1.CommandText = "dbo.[GetState]"
        Me.SqlCommand1.CommandType =
System.Data.CommandType.StoredProcedure
        Me.SqlCommand1.Connection = Me.SqlConnection1
        Me.SqlCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@RETURN_VALUE",
System.Data.SqlDbType.Int, 4,
System.Data.ParameterDirection.ReturnValue, False, CType(0, Byte),
CType(0, Byte), "", System.Data.DataRowVersion.Current, Nothing))
        Me.SqlCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@AccountNo",
System.Data.SqlDbType.NVarChar, 10))
        Me.SqlCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@AccountTypeName",
System.Data.SqlDbType.NVarChar, 50))
        Me.SqlCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@CurrencyName",
System.Data.SqlDbType.NVarChar, 20))

    End Sub
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Label3 As System.Web.UI.WebControls.Label
    Protected WithEvents Label6 As System.Web.UI.WebControls.Label
    Protected WithEvents txtAccountNo As
System.Web.UI.WebControls.TextBox
    Protected WithEvents ddlType As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents Label5 As System.Web.UI.WebControls.Label
    Protected WithEvents Label7 As System.Web.UI.WebControls.Label
    Protected WithEvents Label4 As System.Web.UI.WebControls.Label
    Protected WithEvents ddlCurrency As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents Label8 As System.Web.UI.WebControls.Label
    Protected WithEvents Calendar1 As
System.Web.UI.WebControls.Calendar
    Protected WithEvents Calendar2 As
System.Web.UI.WebControls.Calendar
    Protected WithEvents Label9 As System.Web.UI.WebControls.Label
    Protected WithEvents AccountStateListen As
Microsoft.Speech.Web.UI.Listen
    Protected WithEvents DataGrid1 As
System.Web.UI.WebControls.DataGrid
    Protected WithEvents lbHome As System.Web.UI.WebControls.LinkButton
    Protected WithEvents SqlConnection1 As
System.Data.SqlClient.SqlConnection
    Protected WithEvents SqlCommand1 As
System.Data.SqlClient.SqlCommand
    Protected WithEvents cmdState As System.Web.UI.WebControls.Button

    'NOTE: The following placeholder declaration is required by the Web
Form Designer.
    'Do not delete or move it.
    Private designerPlaceholderDeclaration As System.Object

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()
    End Sub

#End Region

    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Put user code to initialize the page here

        ' txtAccountNo.Text = Session("AccountNo")
        'DataGrid1.DataSource = ""

        If (Not Page.IsPostBack) Then


            Dim cmdCurrency As New SqlCommand("select * from Currency",
SqlConnection1)
            Dim cmdType As New SqlCommand("select * from AccountType",
SqlConnection1)
            SqlConnection1.Open()
            Dim drCurrency As SqlDataReader
            drCurrency = cmdCurrency.ExecuteReader

            Do While drCurrency.Read
                ddlCurrency.Items.Add(drCurrency("CurrencyName"))
            Loop
            drCurrency.Close()

            Dim drType As SqlDataReader

            drType = cmdType.ExecuteReader

            Do While drType.Read
                ddlType.Items.Add(drType("AccountTypename"))
            Loop
            drType.Close()

            SqlConnection1.Close()
        End If
    End Sub

    Protected Sub setDateOnReco(ByVal o As Object, ByVal e As
System.EventArgs)
        'Label6.Text &=
AccountStateListen.RecoResult.SelectSingleNode("Date").SelectSingleNode
("year/text()").Value
        Dim theDate As XmlNode
        Dim tempDate As DateTime
        Dim theMonth, theYear, theDay As Integer


        ddlType.SelectedValue =
AccountStateListen.RecoResult.SelectSingleNode("tt/text()").Value
        ddlCurrency.SelectedValue =
AccountStateListen.RecoResult.SelectSingleNode("cc/text()").Value
        theDate =
AccountStateListen.RecoResult.SelectSingleNode("DateFrom")



        theMonth = theDate.SelectSingleNode("Month/text()").Value
        theDay = theDate.SelectSingleNode("Day/text()").Value
        theYear = theDate.SelectSingleNode("Year/text()").Value



        tempDate = New DateTime(theYear, theMonth, theDay)
        Calendar1.SelectedDate = tempDate
        Calendar1.VisibleDate = tempDate
        theDate =
AccountStateListen.RecoResult.SelectSingleNode("DateFrom")



        theDate =
AccountStateListen.RecoResult.SelectSingleNode("DateTo")

        theMonth = theDate.SelectSingleNode("Month/text()").Value
        theDay = theDate.SelectSingleNode("Day/text()").Value
        theYear = theDate.SelectSingleNode("Year/text()").Value



        tempDate = New DateTime(theYear, theMonth, theDay)

        Calendar2.SelectedDate = tempDate
        Calendar2.VisibleDate = tempDate

        Page.Validate()
    End Sub

    Private Sub cmdState_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cmdState.Click
        Dim dr As SqlDataReader
        SqlCommand1.Parameters("@AccountNo").Value = txtAccountNo.Text
        SqlCommand1.Parameters("@AccountTypeName").Value =
ddlType.SelectedItem.Text
        SqlCommand1.Parameters("@currencyName").Value =
ddlCurrency.SelectedItem.Text
        SqlCommand1.Parameters("@DateFrom").Value =
Calendar1.SelectedDate
        SqlCommand1.Parameters("@DateTo").Value =
Calendar2.SelectedDate


        SqlCommand1.Connection.Open()
        dr = SqlCommand1.ExecuteReader




        DataGrid1.DataSource = dr
        DataGrid1.DataBind()
        dr.Close()
        SqlCommand1.Connection.Close()
    End Sub

    Private Sub lbHome_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles lbHome.Click
        Response.Redirect("wfHome.aspx")
    End Sub
End Class
```

A.1.5 Code Behind of Internal Transfer Page

```
Imports System.Data.SqlClient
Imports System.Web.Security
Imports System.Web.Configuration

Public Class wfChequeOrder
    Inherits System.Web.UI.Page

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.SqlConnection1 = New System.Data.SqlClient.SqlConnection
        '
        'SqlConnection1
        '
        Me.SqlConnection1.ConnectionString = "workstation id=""SUNRISE-
ONAPWVQ"";packet size=4096;user id=sa;data source=""SUNRISE" & _
        "-ONAPWVQ"";persist security info=False;initial
catalog=Bank;password=center"

    End Sub
    Protected WithEvents txtAccountNo As
System.Web.UI.WebControls.TextBox
    Protected WithEvents ddlType As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents Label5 As System.Web.UI.WebControls.Label
    Protected WithEvents Label7 As System.Web.UI.WebControls.Label
    Protected WithEvents Label4 As System.Web.UI.WebControls.Label
    Protected WithEvents ddlCurrency As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents Label3 As System.Web.UI.WebControls.Label
    Protected WithEvents lbHome As System.Web.UI.WebControls.LinkButton
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Label6 As System.Web.UI.WebControls.Label
    Protected WithEvents DropDownList1 As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents Label8 As System.Web.UI.WebControls.Label
    Protected WithEvents TextBox1 As System.Web.UI.WebControls.TextBox
    Protected WithEvents Button1 As System.Web.UI.WebControls.Button
    Protected WithEvents SqlConnection1 As
System.Data.SqlClient.SqlConnection
    Protected WithEvents LinkButton1 As
System.Web.UI.WebControls.LinkButton

    'NOTE: The following placeholder declaration is required by the Web
Form Designer.
    'Do not delete or move it.
    Private designerPlaceholderDeclaration As System.Object

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()
    End Sub

#End Region

    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Put user code to initialize the page here
        txtAccountNo.Text = Session("AccountNo")
        'DataGrid1.DataSource = ""

        If (Not Page.IsPostBack) Then


            Dim cmdCurrency As New SqlCommand("select * from Currency",
SqlConnection1)
            Dim cmdType As New SqlCommand("select * from AccountType",
SqlConnection1)
            SqlConnection1.Open()

            Dim drCurrency As SqlDataReader
            drCurrency = cmdCurrency.ExecuteReader

            Do While drCurrency.Read
                ddlCurrency.Items.Add(drCurrency("CurrencyName"))
            Loop
            drCurrency.Close()

            Dim drType As SqlDataReader
            drType = cmdType.ExecuteReader

            Do While drType.Read
                ddlType.Items.Add(drType("AccountTypename"))
            Loop
            drType.Close()

            SqlConnection1.Close()
        End If
    End Sub

    Private Sub lbHome_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles lbHome.Click
        Response.Redirect("wfHome.aspx")
    End Sub

    Private Sub LinkButton1_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles LinkButton1.Click
        FormsAuthentication.SignOut()
        Response.Redirect("wfLogin.aspx")


    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

    End Sub
End Class
```

A.1.6 Code Behind of Currency Prices Page

```
Imports System.Data.SqlClient
Imports System.Xml

Public Class frmCurrency
    Inherits System.Web.UI.Page


#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.SqlConnection1 = New System.Data.SqlClient.SqlConnection
        '
        'SqlConnection1
        '
        Me.SqlConnection1.ConnectionString = "workstation id=""SUNRISE-
ONAPWVQ"";packet size=4096;user id=sa;data source=""SUNRISE" & _
        "-ONAPWVQ"";persist security info=False;initial
catalog=Bank;password=center"

    End Sub
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Label4 As System.Web.UI.WebControls.Label
    Protected WithEvents ddlFrom As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents Label5 As System.Web.UI.WebControls.Label
    Protected WithEvents ddlTo As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents SqlConnection1 As
System.Data.SqlClient.SqlConnection
    Protected WithEvents lblPrice As System.Web.UI.WebControls.Label
    Protected WithEvents cmdPrice As System.Web.UI.WebControls.Button
    Protected WithEvents Image1 As System.Web.UI.WebControls.Image
    Protected WithEvents CurrencyPriceListen As
Microsoft.Speech.Web.UI.Listen
    Protected WithEvents Label3 As System.Web.UI.WebControls.Label

    'NOTE: The following placeholder declaration is required by the Web
Form Designer.
    'Do not delete or move it.
    Private designerPlaceholderDeclaration As System.Object

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()
    End Sub

#End Region

    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Put user code to initialize the page here
        If (Not Page.IsPostBack) Then

            Dim cmdCurrency As New SqlCommand("select * from Currency",
SqlConnection1)
            'Dim cmdType As New SqlCommand("select * from AccountType",
SqlConnection1)
            SqlConnection1.Open()
            Dim drCurrency As SqlDataReader
            drCurrency = cmdCurrency.ExecuteReader

            Do While drCurrency.Read
                ddlFrom.Items.Add(drCurrency("CurrencyName"))
                ddlTo.Items.Add(drCurrency("CurrencyName"))
            Loop

            drCurrency.Close()
            SqlConnection1.Close()
        End If
    End Sub

    Private Sub cmdPrice_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cmdPrice.Click
        exchange()
    End Sub

    Private Sub exchange()
        Dim cmdCurrency As New SqlCommand("select * from Currency",
SqlConnection1)
        Dim priceFrom, priceTo As Double
        SqlConnection1.Open()

        Dim drCurrency As SqlDataReader
        drCurrency = cmdCurrency.ExecuteReader

        Do While drCurrency.Read
            If drCurrency("CurrencyName") = ddlFrom.SelectedItem.Text
Then
                priceFrom = drCurrency("Price")
            End If
            If drCurrency("CurrencyName") = ddlTo.SelectedItem.Text
Then
                priceTo = drCurrency("Price")
            End If

        Loop
        lblPrice.Text = "price is: " & priceFrom / priceTo

        drCurrency.Close()
        SqlConnection1.Close()
    End Sub

    Protected Sub exchangeOnReco(ByVal o As Object, ByVal e As
System.EventArgs)
```

```
        ddlFrom.SelectedValue =
CurrencyPriceListen.RecoResult.SelectSingleNode("curr1/text()").Value

        ddlTo.SelectedValue =
CurrencyPriceListen.RecoResult.SelectSingleNode("curr2/text()").Value

        exchange()

    End Sub

End Class
```

A.1.7 Code Behind of Check Book Order Page

```
Imports System.Data.SqlClient
Imports System.Web.Security
Imports System.Web.Configuration

Public Class wfChequeOrder
    Inherits System.Web.UI.Page

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
        Me.SqlConnection1 = New System.Data.SqlClient.SqlConnection
        Me.SqlCommand1 = New System.Data.SqlClient.SqlCommand
        '
        'SqlConnection1
        '
        Me.SqlConnection1.ConnectionString = "workstation id=""SUNRISE-
ONAPWVQ"";packet size=4096;user id=sa;data source=""SUNRISE" & _
        "-ONAPWVQ"";persist security info=False;initial catalog=Bank"
        '
        'SqlCommand1
        '
        Me.SqlCommand1.CommandText = "dbo.[ChequeOrder]"
        Me.SqlCommand1.CommandType =
System.Data.CommandType.StoredProcedure
        Me.SqlCommand1.Connection = Me.SqlConnection1
        Me.SqlCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@RETURN_VALUE",
System.Data.SqlDbType.Int, 4,
System.Data.ParameterDirection.ReturnValue, False, CType(0, Byte),
CType(0, Byte), "", System.Data.DataRowVersion.Current, Nothing))
        Me.SqlCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@AccountNo",
System.Data.SqlDbType.Decimal, 9, System.Data.ParameterDirection.Input,
False, CType(18, Byte), CType(0, Byte), "",
System.Data.DataRowVersion.Current, Nothing))
        Me.SqlCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@AccountTypeName",
System.Data.SqlDbType.NVarChar, 50))
        Me.SqlCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@CurrencyName",
System.Data.SqlDbType.NVarChar, 50))
        Me.SqlCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@orderNo",
System.Data.SqlDbType.Decimal, 9, System.Data.ParameterDirection.Input,
False, CType(18, Byte), CType(0, Byte), "",
System.Data.DataRowVersion.Current, Nothing))
        Me.SqlCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@chequeLang",
System.Data.SqlDbType.NVarChar, 50))
        Me.SqlCommand1.Parameters.Add(New
System.Data.SqlClient.SqlParameter("@NoOfCheques",
System.Data.SqlDbType.Decimal, 9, System.Data.ParameterDirection.Input,
False, CType(18, Byte), CType(0, Byte), "",
System.Data.DataRowVersion.Current, Nothing))

    End Sub
    Protected WithEvents txtAccountNo As
System.Web.UI.WebControls.TextBox
    Protected WithEvents ddlType As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents Label5 As System.Web.UI.WebControls.Label
    Protected WithEvents Label7 As System.Web.UI.WebControls.Label
    Protected WithEvents Label4 As System.Web.UI.WebControls.Label
    Protected WithEvents ddlCurrency As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents Label3 As System.Web.UI.WebControls.Label
    Protected WithEvents lbHome As System.Web.UI.WebControls.LinkButton
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents Label6 As System.Web.UI.WebControls.Label
    Protected WithEvents Label8 As System.Web.UI.WebControls.Label
    Protected WithEvents LinkButton1 As
System.Web.UI.WebControls.LinkButton
    Protected WithEvents SqlConnection1 As
System.Data.SqlClient.SqlConnection
    Protected WithEvents SqlCommand1 As
System.Data.SqlClient.SqlCommand
    Protected WithEvents cmdOrder As System.Web.UI.WebControls.Button
    Protected WithEvents txtNoOfCheques As
System.Web.UI.WebControls.TextBox
    Protected WithEvents ddlLang As
System.Web.UI.WebControls.DropDownList
```

```
    'NOTE: The following placeholder declaration is required by the Web
Form Designer.
    'Do not delete or move it.
    Private designerPlaceholderDeclaration As System.Object

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()
    End Sub

#End Region

    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Put user code to initialize the page here
        txtAccountNo.Text = Session("AccountNo")
        'DataGrid1.DataSource = ""

        If (Not Page.IsPostBack) Then


            Dim cmdCurrency As New SqlCommand("select * from Currency",
SqlConnection1)
            Dim cmdType As New SqlCommand("select * from AccountType",
SqlConnection1)
            SqlConnection1.Open()

            Dim drCurrency As SqlDataReader
            drCurrency = cmdCurrency.ExecuteReader

            Do While drCurrency.Read
                ddlCurrency.Items.Add(drCurrency("CurrencyName"))
            Loop
            drCurrency.Close()

            Dim drType As SqlDataReader

            drType = cmdType.ExecuteReader

            Do While drType.Read
                ddlType.Items.Add(drType("AccountTypename"))
            Loop
            drType.Close()

            SqlConnection1.Close()
        End If
    End Sub
    Private Sub lbHome_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles lbHome.Click
        Response.Redirect("wfHome.aspx")
    End Sub

    Private Sub LinkButton1_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles LinkButton1.Click
        FormsAuthentication.SignOut()
        Response.Redirect("wfLogin.aspx")


    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)

    End Sub

    Private Sub cmdOrder_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cmdOrder.Click
        SqlCommand1.Parameters("@AccountNo").Value = txtAccountNo.Text
        SqlCommand1.Parameters("@AccountTypeName").Value =
ddlType.SelectedItem.Text
        SqlCommand1.Parameters("@CurrencyName").Value =
ddlCurrency.SelectedItem.Text
        SqlCommand1.Parameters("@NoOfCheques").Value =
txtNoOfCheques.Text
        SqlCommand1.Parameters("@AccountNo").Value =
ddlLang.SelectedItem.Text

        SqlConnection1.Open()
        SqlCommand1.ExecuteNonQuery()
        SqlConnection1.Close()

    End Sub
End Class
```

Appendix B

Using Grammars

Using Grammar Elements to Recognize Speech

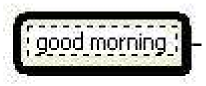
Grammar rules should account for all possible user speech patterns in a particular speech application. Use the elements in Speech Grammar Editor to design grammar rules that define user speech patterns.

Recognizing Words and Phrases

Use a Phrase element to specify the text of a single word, phrase, or complete sentence that the speech recognition engine recognizes. Phrase elements are the building blocks of grammar rules.

The Phrase element represents an item XML markup element in the XML text of a .grxml file.

The following snippet illustrates using one Phrase element in Speech Grammar Editor to represent the phrase "good morning."

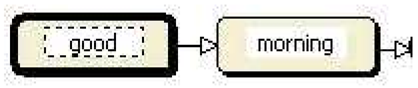


The following snippet illustrates the corresponding XML markup.

```
<rule id="PhraseElement">
```

```
    <item>good morning</item>
</rule>
```

The following snippet illustrates using two Phrase elements, one for each word, in Speech Grammar Editor to represent the phrase "good morning."



The following snippet illustrates the corresponding XML markup.

```
<rule id="PhraseElement" scope="private">
    <item>good</item>
    <item>morning</item>
</rule>
```

Note When entering text values for Phrase elements, enter text without using quotes. Text enclosed by quotes is not recognized by the speech recognition engine.
Note No element in Speech Grammar Editor represents a rule XML markup element. Each window in Rule Editor represents a rule XML markup element.

When the speech recognition engine combines with either of the previous rules to recognize the phrase "good morning", the recognition engine sends the following SML results to an application.

```
<SML text="good morning" utteranceConfidence="1.000">
    good morning
</SML>
```

Customizing Pronunciation

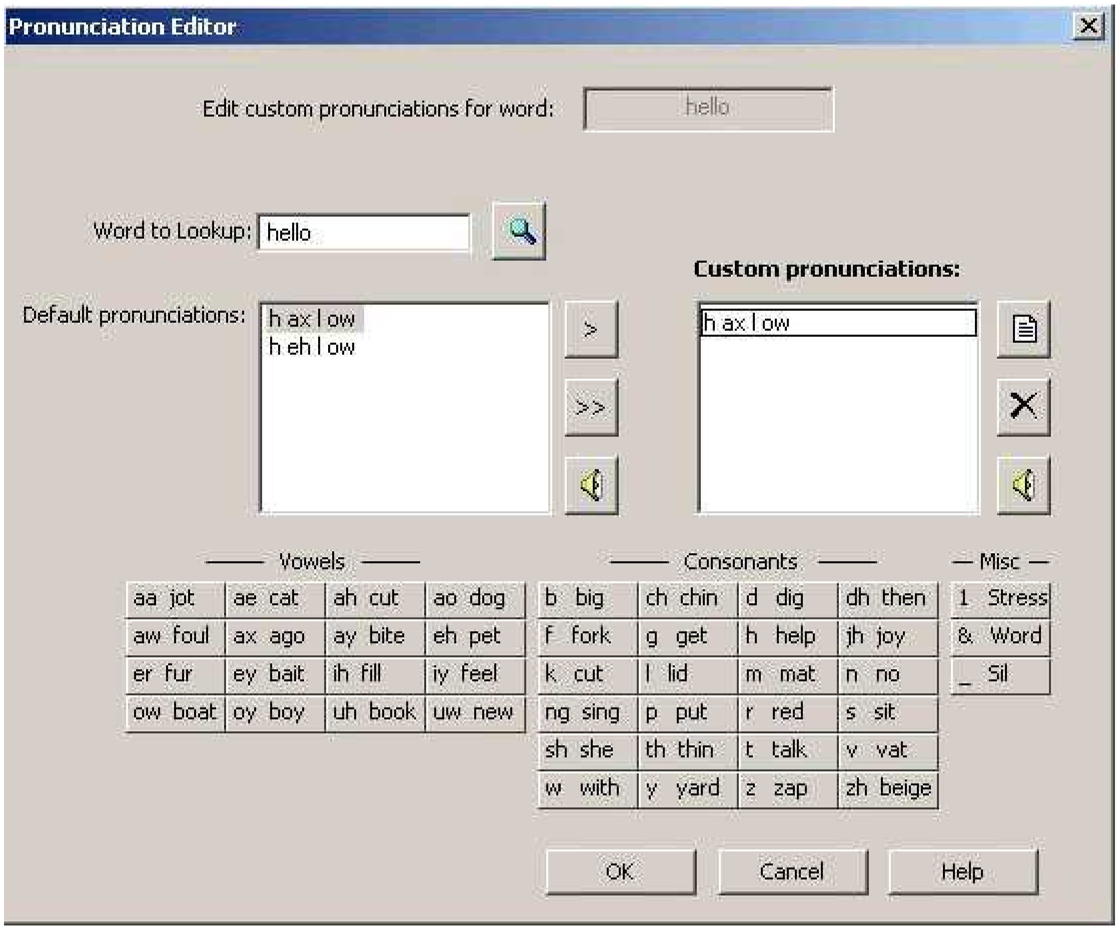
Speech Grammar Editor provides Pronunciation Editor, which enables grammar authors to define custom pronunciations of single-word Phrase elements.

By default, Speech API (SAPI) provides default pronunciations for all words. These default pronunciations are comprised of phonemes, which are abstract categories of speech sounds (vowels and consonants) grouped together to create words.

For example, SAPI provides two default pronunciations of the word "hello." Each group of sounds in the following snippet, separated by spaces, represents a phoneme.

```
h ax l ow
h eh l ow
```

Use Pronunciation Editor to create custom pronunciations of single words.



Edit custom pronunciations for word: hello

Word to Lookup: hello

Default pronunciations:
h ax l ow
h eh l ow

Custom pronunciations:
h ax l ow

Vowels

aa jot	ae cat	ah cut	ao dog
aw foul	ax ago	ay bite	eh pet
er fur	ey bait	ih fill	iy feel
ow boat	oy boy	uh book	uw new

Consonants

b big	ch chin	d dig	dh then
f fork	g get	h help	jh joy
k cut	l lid	m mat	n no
ng sing	p put	r red	s sit
sh she	th thin	t talk	v vat
w with	y yard	z zap	zh beige

Misc

1 Stress
& Word
_ Sil

Note Individual phonemes in Pronunciation Editor are disabled until a pronunciation in the Custom pronunciations pane is editable.

To define a custom pronunciation

1. Right-click a Phrase element on the Rule Editor window that contains a single word, and on the shortcut menu, click Properties.
2. On the Properties window, click the Pronunciation field, and then click the ellipsis (...) button to open Pronunciation Editor.

3. On the Word to lookup text box, click Lookup the pronunciations to display SAPI default pronunciations for the word in the Default pronunciations pane, and then do one of the following:
    - On the Default pronunciations pane, click a default pronunciation, and then click Play selected pronunciation to hear a text-to-speech (TTS) representation of the selected pronunciation.
    - On the Default pronunciations pane, click a default pronunciation, and then click Insert selected pronunciation to add the selected default pronunciation to the Custom pronunciations pane, where authors can modify its pronunciation.
    - On the Default pronunciations pane, click Copy all lookup pronunciations to add all of the default pronunciations to the Custom pronunciations pane, where authors can modify all of their pronunciations.
4. Depending on the action taken in the previous step, on the Custom pronunciations pane, do one of the following.
    - Click a custom pronunciation to enable the Phoneme Palette.
    - Click Add new custom pronunciation to enable the Phoneme Palette.
5. Click phonemes under the Vowels, Consonants, or Misc headings to create custom pronunciations.
6. To hear a TTS representation of the selected pronunciation at any time, on the Custom pronunciations pane, click a custom pronunciation, and then click Play selected pronunciation.
7. Click OK to save changes and close Pronunciation Editor.

When grammar authors add a custom pronunciation to a Phrase element, the Phrase element represents a token XML markup element with a pron attribute in the XML text of a .grxml file instead of the default item XML markup element.

```
<token sapi:pron="h eh l ow 1">hello</token>
```

Custom phonemes appear in the Phrase element's Properties window on the Pronunciation field.

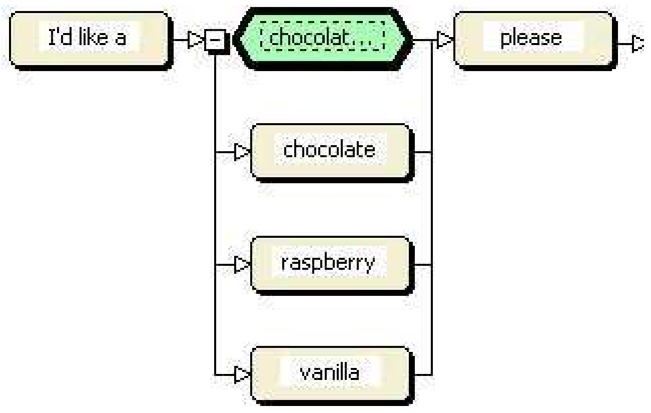
Recognizing Choices From a List

Use List elements to specify a set of alternative words or phrases that a user might speak to an application. When a user says one of the words or phrases in the list, the recognition engine recognizes that word or phrase.

When a grammar author drags a List element onto the Rule Editor window, a child Phrase element appears underneath the List element. Although this is the default behavior, List elements may contain any type of child elements, including Group elements and other List elements.

The List element represents a one-of XML markup element in the XML text of a .grxml file.

The following snippet illustrates using a List element on Speech Grammar Editor to enclose a list of flavor choices—chocolate, raspberry, and vanilla. Three Phrase elements specify the flavor choices, using one Phrase element per choice.



The following snippet illustrates the corresponding XML markup.

```
<rule id="ListExample" scope="private">
    <item>I'd like a</item>
    <one-of>
        <item>chocolate</item>
        <item>raspberry</item>
        <item>vanilla</item>
    </one-of>
    <item>please</item>
</rule>
```

When the recognition engine combines with this rule to recognize the phrase "I'd like a chocolate please", the recognition engine sends the following SML results to an application.

```
<<SML text="I'd like a chocolate please" utteranceConfidence="1.000">
    I'd like a chocolate please
</SML>
```

Note All child elements of the containing List element inherit the properties of the parent List element.

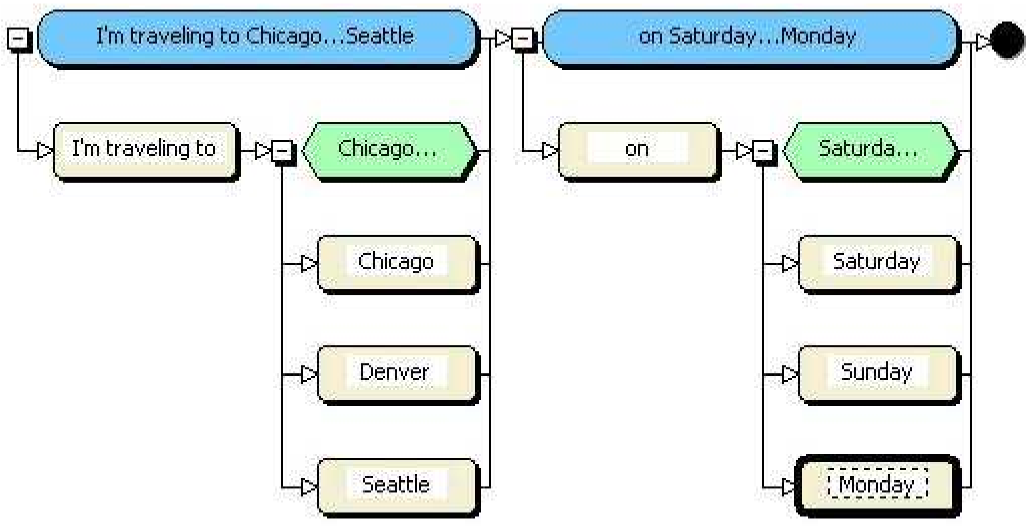
Grouping Related Information

Use the Group element to specify that a group of elements are related, and also to assign properties to a group of elements. For example, if a grammar author sets the Max Repeat property on a Group element, the value of that property also applies to the elements contained in the group. The recognition engine recognizes each item in the group.

When a grammar author drags a Group element onto the Rule Editor window, a child Phrase element appears underneath the Group element. Although this action is the default behavior, Group elements may contain any type of child elements, including List elements and other Group elements.

The Group element represents an item XML markup element in the XML text of a .grxml file. In contrast to the Phrase element, which encloses individual words or phrases in separate item XML markup elements, the Group element encloses all elements in the group with a single item XML markup element.

The following snippet illustrates using two Group elements in Speech Grammar Editor—one to group information about the destination city, and one to group information about travel dates.



The following snippet illustrates the corresponding XML markup.

```
<rule id="GroupExample" scope="private">
    <item>
        <item>I'm traveling to</item>
        <one-of>
            <item>Chicago</item>
            <item>Denver</item>
            <item>Seattle</item>
        <one-of>
    </item>
    <item>
        <item>on</item>
        <one-of>
            <item>Saturday</item>
            <item>Sunday</item>
            <item>Monday</item>
        <one-of>
    </item>
</rule>
```

When the recognition engine combines with this rule to recognize the phrase "I'm traveling to Seattle on Monday", the recognition engine sends the following SML results to an application.

```
<SML text="I'm traveling to Seattle on Monday"
utteranceConfidence="1.000">
    I'm traveling to Seattle on Monday
</SML>
```

Note Although there is no specific World Wide Web Consortium Speech Recognition Grammar Specification Version 1.0 (W3C SRGS) element corresponding to a Group element, using this element does not break W3C SRGS compliance.

Referencing Other Rules

Use the RuleRef element to reference another rule, either from the same grammar file or from an external grammar file. Use RuleRef elements to reuse preexisting component grammars. For example, the cmnrules.cfg file installed with the Microsoft Speech Application SDK Version 1.0 Beta 3 (SASDK) contains preexisting rules that define common items such as dates, credit card numbers, telephone numbers, and other items.

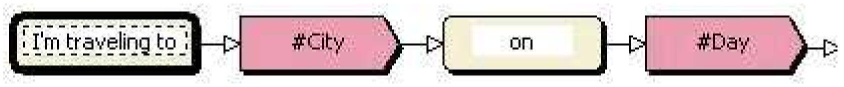
Note If the Microsoft Speech Application SDK Version 1.0 Beta 3 (SASDK) is installed to the default location, cmnrules.cfg can be found at: %SystemDrive%\Inetpub\wwwroot\aspnet_speech\%Version%\client_script\1033

To specify the target rule of a RuleRef

- Click a RuleRef element on the Rule Editor window, and enter a value for the URI property in the Properties window.
- - or -
- Right-click a RuleRef element, on the shortcut menu point to Set Target Rule, and then on the shortcut sub-menu click a rule name, or click Other, to reference a rule within the same grammar file. If no rule names appear on the shortcut sub-menu, no other rules exist within the grammar file. Click Browse to find a rule in an external grammar.

The RuleRef element represents a ruleref XML markup element in the XML text of a .grxml file.

The following snippet illustrates using two RuleRef elements in Speech Grammar Editor to reference two rules, City and Day, in the same grammar file.



The following snippet illustrates the corresponding XML markup.

```
<rule id="RuleRefExample" scope="private">
    <item>I'm traveling to</item>
    <ruleref uri="#City" type="application/srgs+xml"/>
    <item>on</item>
    <ruleref uri="#Day" type="application/srgs+xml"/>
</rule>

<rule id="City" scope="private">
    <one-of>
        <item>Chicago</item>
        <item>Denver</item>
        <item>Seattle</item>
    </one-of>
</rule>

<rule id="Day" scope="private">
    <one-of>
        <item>Saturday</item>
        <item>Sunday</item>
        <item>Monday</item>
    <one-of>
</rule>
```

When the recognition engine combines with these rules to recognize the phrase "I'm traveling to Seattle on Monday", the recognition engine sends the following SML results to an application.

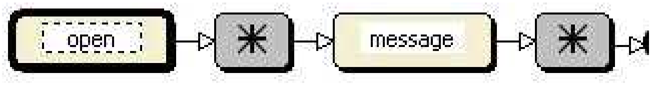
```
<SML text="I'm traveling to Seattle on Monday"
utteranceConfidence="1.000">
    I'm traveling to Seattle on Monday
</SML>
```

Ignoring Irrelevant Words

Use the Wildcard element if the recognition engine recognizes a phrase but ignores irrelevant words. For example, it may be necessary to recognize the phrase "open message." Recognition should not fail if a user says *open my message*, *open the message*, or *open the message please*. Use the Wildcard element to discard the unnecessary words.

The Wildcard element represents a ruleref XML markup element with its special attribute set to GARBAGE in the XML text of a .grxml file.

The following snippet illustrates using two Wildcard elements in Speech Grammar Editor. These elements enable the recognition engine to recognize speech if a user says *open my message*, *open the message*, *open the message please*, or any number of other responses.



The following snippet illustrates the corresponding XML markup.

```
<rule id="WildcardExample" scope="private">
    <item>open</item>
    <ruleref special="GARBAGE"/>
    <item>message</item>
    <ruleref special="GARBAGE"/>
</rule>
```

When the recognition engine combines with this rule to recognize the phrase "open my message please", the recognition engine sends the following SML results to an application.

```
<SML text="open ... message ..." utteranceConfidence="1.000">
    open ... message ...
</SML>
```

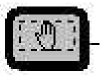
Note The ellipses are actually part of the SML results. They represent the unnecessary spoken words.

Disabling Recognition

Use the Halt element to disable recognition of any recognition path that contains this element. This element enables grammar authors to isolate particular recognition paths and rules for testing purposes during development. If Speech Grammar Editor activates a recognition path or rule containing the Halt element, recognition fails.

The Halt element represents a ruleref XML markup element with its special attribute set to VOID in the XML text of a .grxml file.

The following snippet illustrates using a Halt element on the Speech Grammar Editor.



The following snippet illustrates the corresponding XML markup.

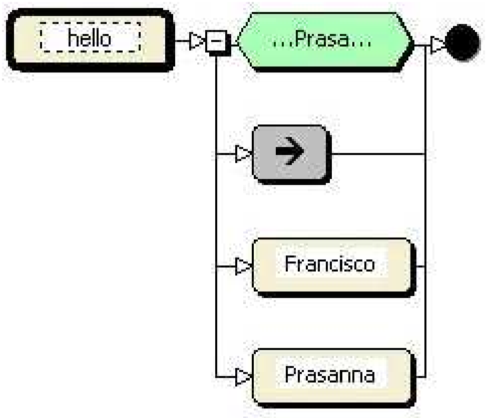
```
<rule id="HaltExample" scope="private">
    <ruleref special="VOID"/>
</rule>
```

Ignoring Recognition

Use the Skip element to treat a recognition path as optional. Using the Skip element on a recognition path has the same effect as specifying the Min Repeat and Max Repeat properties on that path.

The Skip element represents a ruleref XML markup element with its special attribute set to NULL in the XML text of a .grxml file.

The following snippet illustrates a Skip element within a List element. The Skip element makes the List optional, so the recognition engine would recognize both *hello* or *hello Prasanna*.



The following snippet illustrates the corresponding XML markup.

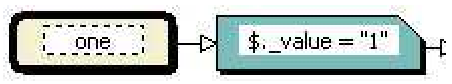
```
<rule id="SkipExample" scope="public">
    <item>hello</item>
    <one-of>
        <item>
            <ruleref special="NULL"/>
        </item>
        <item>Francisco</item>
        <item>Prasanna</item>
    </one-of>
</rule>
```

Adding Semantic Interpretation Information

Use the Script Tag element to associate other rule elements with property values and scripts that manipulate the Semantic Markup Language (SML) text results returned by the recognition engine and sent to an application.

The Script Tag element represents a tag XML markup element in the XML text of a .grxml file.

The following snippet illustrates using a Script Tag element in Speech Grammar Editor.



The following snippet illustrates the corresponding XML markup.

```
<rule id="Tickets" scope="private">
    <item>one</item>
    <tag>$._value = "1"</tag>
</rule>
```