Santa Clara University
DEPARTMENT of COMPUTER ENGINEERING

Date: June 13, 2006

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Anders Hur and Wilson Le

ENTITLED

Centralized Phonebook Database

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER ENGINEERING

_____
THESIS ADVISOR

_____
DEPARTMENTCHAIR

Centralized Phonebook Database

by

Anders Hur and Wilson Le

SENIOR DESIGN PROJECT REPORT

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Engineering
School of Engineering
Santa Clara University

Santa Clara, California

June 13, 2006

Abstract

People have become highly dependent on their cell phone phonebooks; so much so that most people do not even try to memorize phone numbers anymore. If a person does not have access to their cell phone, they are not able to get to phone numbers that they might want to call. We created a system that allows a person to access their phonebook virtually anywhere in the world at anytime.

In the Centralized Phonebook Database, a person uploads their phonebook from their cell phone to a server. The phonebook is then stored in the server and can be accessed in two different ways. The first is to access it through voicemail. The use of voicemail allows a person to access their phonebook from any location that has a telephone at anytime. The second method of phonebook retrieval is through a website. This allows people to access their phonebook from any computer connected to the internet. The entire system consists of a web server, database, and voicemail server.

Keywords: backup/recovery, phonebook, database, cell phone, voicemail

Acknowledgements

We would like to thank our senior design project advisor Dr. Dan Lewis. He helped us develop our ideas and implement them. His guidance was essential for the success of this project. Sumit Naiksatam helped us with our database and website. His knowledge on Java and databases played a vital role in the success of our project. Tan Trieu also provided helpful ideas and suggestions.

We would also like to thank Santa Clara University and the Computer Engineering Department. Without their resources and teachings this project would not have been possible.

Finally, we would like to thank our families' and God, for without their love, support, and guidance we would not be where we are today.

Table of Contents

List of Figures

Page

7. Introduction

7.1 Motivation

A major issue facing cell phone users is the absence of their cell phones when they are needed most. This occurs when a person forgets, damages, or loses their cell phone. If a person does not have their cell phone they cannot use the functionality that people take for granted when they have their cell phones.

One of these features is the use of their phonebook. People are highly dependent on their cell phone phonebooks; so much so that most people do not even try to memorize phone numbers anymore. These phonebooks are also critical because of the increased number of phone numbers that people need to store; such as home, work, and cell phone.

If a person forgets their cell phone at home, they are not able to get to phone numbers that they might want to call during the day. Furthermore, if a person breaks or loses their cell phone permanently, this crucial information is lost forever. Reacquiring hundreds of phone numbers is very tedious and often an impossible task. Our project aims to eliminate these problems for cell phone users.

7.2 Background

Currently there is one method to help prevent complete loss of a phonebook. Companies (e.g. DataPilot) sell USB cords with software to upload a phonebook to a personal computer. While this helps in the case of a broken or lost cell phone, it does not address the problem completely. A person can only use the stored phonebook to look up numbers at their personal computer. However, this system does not allow for on-demand access of phone numbers when they are away from their computer.

7.3 Solution

Current systems only allow people to access their phonebooks from their personal computer. Our system solves this problem. When a person forgets, breaks, or loses their cell phone, they will be able to access their phonebook from anywhere in the world at anytime.

In our system, a person uploads their phonebook from their cell phone to a server. The phonebook is then stored in the server and can be accessed in two different ways. The first is to access it through voicemail. The use of voicemail allows a person to access their phonebook from any location that has a telephone at anytime. The second method of phonebook retrieval is through a website. This allows people to access their phonebook from any computer connected to the internet.

The four basic parts of the system and its use are listed below:

Cell phone: Contains the local phonebook that a person adds numbers into and uses to find numbers and call people. The phonebook is uploaded to the servers via computer using third party software and CGI.

Servers: Phonebooks are stored on both a HTTP server and an Oracle database. The phonebooks are saved on the HTTP server as Common Separated Value files. The information in these files is inserted into the Oracle database during upload. The HTTP server is accessible from anywhere in the world. The voicemail system and web interface access the information stored on the servers.

Voicemail system: The best way to describe the voicemail system is to give an example of its use:

A person calls into their voicemail when they have forgotten or lost their cell phone and want to get a number from their phonebook. They enter their PIN number to access the voicemail. They then use the keypad to enter the name of the person they want to call using "T9" style text entry. The server deciphers the entered name and reads back the phone number of the desired person. From here the person can write down the phone number.

Web Interface: The web interface is a supplementary feature of the overall system. While the voicemail system provides the key functionality, the web interface allows a person to access their phonebook online. If a person is near a computer when they need to access their phonebook they are able to go to the website and access their

phonebook. This might be more convenient and for computer savvy users, is probably a simpler way to access their phonebook. However, the web interface does not have the versatility that is inherent with voicemail because it can only be accessed at a computer rather than from any phone.

7.4 Conceptual Model

Figure 7.1 Conceptual Model

8. Requirements

8.1 Non-functional

- Phonebooks must be available at all times and accessible on demand
- The upload process should support most, if not all cell phones.
- The entire process must be easy to use for all types of users.
- Minimal additional knowledge should be required of system users.

8.2 Functional

Software

Database:

- Must be able to store phonebooks in a format that is easy to create and retrieve

Website:

- Must authenticate users by comparing stored user phone number and pin with entered phone number and pin
- Must be able to upload phonebook from users computer to server and database
- Must be able to access database to retrieve stored phonebooks
- Must display user's phonebook
- Ability to search phonebook for contact's information

Voicemail:

- Must be able to detect and answer calls and then detect end of calls and hang up
- Must be able to create customized menu trees
- Must be able to receive user entered inputs (DTMF tones)
- Must be able to respond with proper response message

- Must execute program to download phonebook file from web server to voicemail server
- Must extract requested data from phonebook file

Hardware

Modem

- Must be able to interact with voicemail software
- Must be able to detect DTMF tones
- Must be able to transmit voice response at audible sound levels
- Must be able to detect and answer calls and then detect end of calls and hang up

Upload Cable

- Must provide a physical connection between cell phone and computer
- Must be able to upload phonebook from cell phone to computer

9. Use Cases

In this chapter we utilize “use cases” to illustrate the functionality of the Centralized Phonebook Database as seen by the designer. Use cases are a widely used and useful method to discover and understand functional requirements that indicate what the system will do. Use cases are also a mechanism to help keep it simple and understandable for all stakeholders. They are a model for describing the system’s functionality and environment. In essence, use cases are a way of discovering and recording functional requirements by writing stories of the use a system to help fulfill stakeholders’ goals.

There are six parts to a use case:

- **Primary Actor.** The primary actor is the entity that performs the majority of the actions that are critical to the use case. Examples of actors are a person, computer system, or organization.
- **Stakeholders and Interest.** Stakeholders are the entities that have an interest in the success of the scenario. This section includes a list of stakeholders and their interests in the scenario.
- **Preconditions.** These are the conditions that need to be fulfilled before the scenario in the use case begins. These conditions are not tested in the scenario, instead, they are assumed to be true when the scenario starts.
- **Success Guarantee (post-condition).** This is the result of a successfully completed scenario. The guarantee should meet all the needs of the stakeholders.
- **Main Success Scenario.** The list of steps that describes the typical success path that satisfies the interest of the stakeholders.
- **Extension (or Alternative Flow).** Indicates other scenarios or branches for both success and failure from the main success scenario.

We created three use cases for the CPD. The three use cases are: Phonebook Upload to Server, Phonebook Access via Voice Mail, and Phonebook Access via Webpage Interface. We chose these three scenarios because they are the three main functionalities of the CPD system that involve the cell phone user. All scenarios have the cell phone

user as the Primary Actor and scenarios that do not involve the cell phone user are not included.

Use Case 1: Phonebook Upload to Server

This case describes what happens when the cell phone user wants to upload their phonebook to the server. Before a user can start uploading their phonebook to the CPD server, they must first upload their phonebook to their computer in Comma Separated Values (CSV) format. The user must then open a web browser and proceed to the CPD website (www.cpd.com). Once the user logs into the website using their cell phone number and pin, the user selects the "Browse" button and finds the location of their phonebook information. Finally, the user clicks on the "Upload" button.

At this point the server software will extract the phonebook information from the CSV file and upload the information to the database. The server will also store a copy of the phonebook information in CSV file format.

This scenario is modeled in the following use case:

Primary Actor: Cell phone user

Stakeholders and Interest:

- Cell phone user: Wants to quickly and accurately upload phonebook to centralized server.
- Server: Wants phonebooks in a standardized form from cell phone user's computer.
- System Administrator: Wants secure and trouble free upload of phonebooks from user's computer to server.
- Company: Wants successful upload of phonebook as part of fulfilling the cell phone user's desire for a centralized phonebook.

Preconditions: User has successfully uploaded phonebook to computer in CSV format and is already logged into the CPD website.

Success Guarantee (post-conditions): Phonebook is stored in Server.

Main Success Scenario:

1. Cell phone users selects "Browse" button to select location of phonebook information.
2. Cell phone users selects location of phonebook information.

3. Cell phone users selects “Upload Phonebook” button.

4. Server Software extracts phonebook information from CSV file.

5. Software uploads the phonebook information to the database.

6. Software also stores a copy of the phonebook information in CSV file format on the server.

Extension (or Alternative Flows):

2a. Cell phone user selects the wrong location of phonebook information.

1. Cell phone user clicks on “Browse” button
2. Cell phone user selects a new location of phonebook information.

4a. Software cannot extract phonebook information from CSV file.

1. Software asks cell phone user to re-upload phonebook from cell phone and make sure that it is uploaded in CSV file format.

6a. Server is unable to store phonebook information in database.

1. Software will return with an “Unsuccessful Upload” message.

Use Case 2: Phonebook Access via Voice Mail.

This case describes what happens when the cell phone user wants to access their phonebook through voicemail. Before the user can access their phonebook through their voicemail, they must have first uploaded it to the server, have a voicemail PIN, and know how to use T9 keypad entry.

To access a phonebook via voicemail the user must call into their voicemail and enter their PIN when prompted. The voicemail system will authenticate the user and prompt the user to enter the desired contact’s name. The user will then enter the contacts name using their keypad.

Upon entry, the system will identify possible matches and read back a list of the contacts. The user will select the number of the desired contact and the system will read back the phone number of the contact. When the system is done reading back the phone number it will give the user three options; repeat the previous number, search for another contact, or hang up. The user will pick the desired option and continue the process.

This scenario is modeled in the following use case:

Primary Actor: Cell phone user

Stakeholders and Interest:

- Cell phone user: Wants to easily and quickly access phone numbers in phonebook via voice mail.
- Voice Mail Software: Want to accurately identify the cell phone user. Wants to identify which phone number the user desires. Wants to access the phone number that the cell phone user wants. Wants to read back number to the cell phone user.
- Server: Wants to successfully access the phone number that is need by the cell phone user and relay it to the cell phone user via voicemail software.
- System Administrator: Wants a secure and trouble free access of phonebook by the user via the cell phone user's voice mail.
- Company: Wants successful access of phone number as part of fulfilling the cell phone user's desire for a centralized phonebook.

Preconditions: Phonebook has already been uploaded to the Server, user has already created a PIN for voice mail, and user knows how to use T9 style keypad entry to enter name of desired phonebook entry.

Success Guarantee (post-conditions): User successfully accesses desired contact's phone number.

Main Success Scenario:

1. Cell phone user calls into his or hers voice mail system.
2. The voice mail system prompts the user to enter a PIN.
3. Cell phone user identifies his or herself by entering a PIN.
4. Upon successful login, voice mail system prompts the user to enter contact's name using key pad.
5. User enters the name of the contact using the keypad.
6. System identifies a list of possible matches from the names in the database.
7. System reads back the list of possible contacts to the user.
8. User selects the desired contact's name from the list.
9. System reads backs the number of the contact.

10. System prompts user if he or she wants to repeat the number, search for another contact's phone number, or is done.

11a. If user chooses to repeat the previous number:

12. System will go back to step 7 and reread number of contact

11b. If user chooses to search for another contact's phone number:

12. The systems goes back to step 4.

11c. If user is done, he or she hangs up.

Extension (or Alternative Flows):

3a. User enters incorrect PIN

1. Voice mail system prompts for reentry of PIN
2. User reenters PIN.

2a. If user reenters pin incorrectly 2 times, disconnect call.

5a. User mistakenly enters incorrect key entry

1. User presses # to submit mistaken entry
2. Voice mail recognizes incorrect entry
3. Voice mail prompts user to reenter name of contact

6a. System cannot find name of contact in phonebook

1. System asks user to reenter name of contact

1a. System still cannot find name of contact

1. System tells user that contact name cannot be found

11a. User enters unknown option

1. System repeats list of options

<u>Use Case 3</u>: Phonebook Access via Webpage Interface.

This case describes what happens when the cell phone user wants to access their phonebook through the webpage interface. Before the user can access their phonebook through the webpage, they must have first uploaded it to the server and then access the CPD webpage.

Once at the CPD webpage, the user will enter their cell phone number and voicemail PIN to log in. The webpage will authenticate the user and display the user's

entire phonebook. There are then two ways the user can search for a desired contact's phone number. The first way is to click on the letter to the right of "LINKS" which corresponds to the first letter of the first name of the contact you are trying to find. This link will scroll the browser to the section that contains names beginning with that letter. The second way to search the phonebook is to type the name of the person whose phone number is desired in the search text box and click on the "Find Name in Page" button. The web browser will then scroll down to the matching contact's name and will highlight the name.

This scenario is modeled in the following use case:

Primary Actor: Cell phone user

Stakeholders and Interest:

- Cell phone user: Wants to quickly and easily retrieve the phone number of the desired contact via the web interface.
- Web interface: Wants to successfully identify the user through a login screen. Wants to retrieve his or her stored phonebook from the database. Wants to search for the desired contact's phonebook. Wants to clearly display the desired phone number for the user.
- Server/Database: Wants allow the web interface to easily access its phonebook database.
- System Administrator: Wants a secure and smooth access of phonebook by the cell phone user via the web interface.
- Company: Wants successful retrieval of a phone number by the cell phone user via the web interface as part of the user's desire for a centralized phonebook.

Preconditions: Phonebook has already been uploaded to the Server and the cell phone user is at web interface.

Success Guarantee (post-conditions): User is able to login successfully, view entire phonebook, and obtain the phone number of the desired contact.

Main Success Scenario:

1. Cell phone user enters cell phone number and PIN.

2. Web interface displays user's phonebook

3. User can use one of the two methods available to search for a desired contact's phone number.

Extension (or Alternative Flows):

1a. Cell phone user enters the wrong password and or user name.

1. Web interface returns the message "Unable to Login."
2. Web interface allows for reentry of user name and password.

2a. User has not uploaded a phonebook

1. Web interface displays "No phonebook uploaded. Please upload phonebook."

3a. User uses letter links to search for a desired contact's phone number.

1. User clicks on the letter to the right of "LINKS" which corresponds to the first letter of the first name of the contact you are trying to find.
2. Web browser will scroll down to the section that contains names beginning with that letter.

3b. User uses search box to search for a desired contact's phone number

1. User types in the name of the desired contact in the search box.
2. User selects the "Find Name in Page" button.
3. Web browser scrolls down to matching contact's name and highlights it.

10. Sequence Diagrams

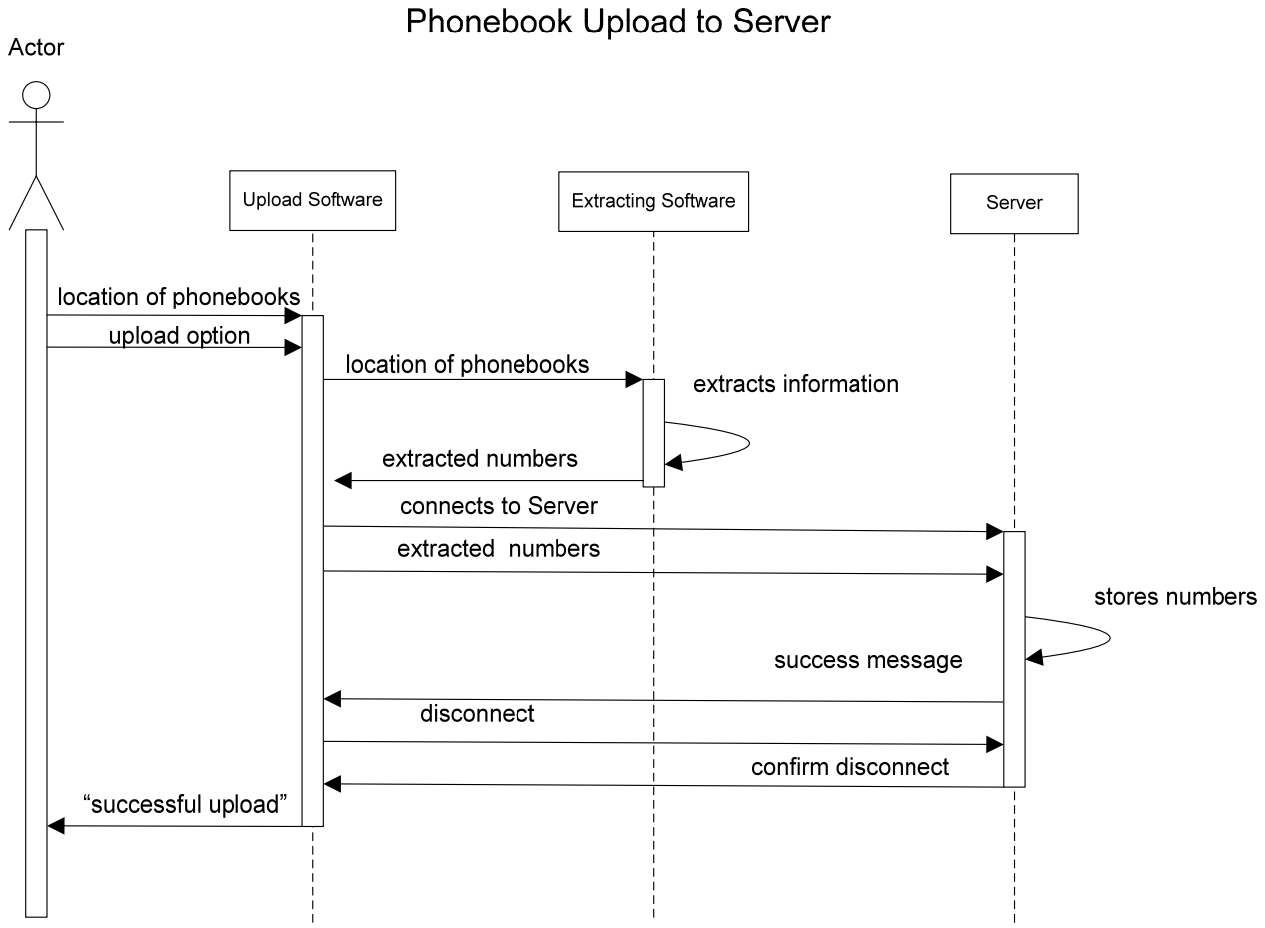


Figure 10.1 Phonebook Upload to Server

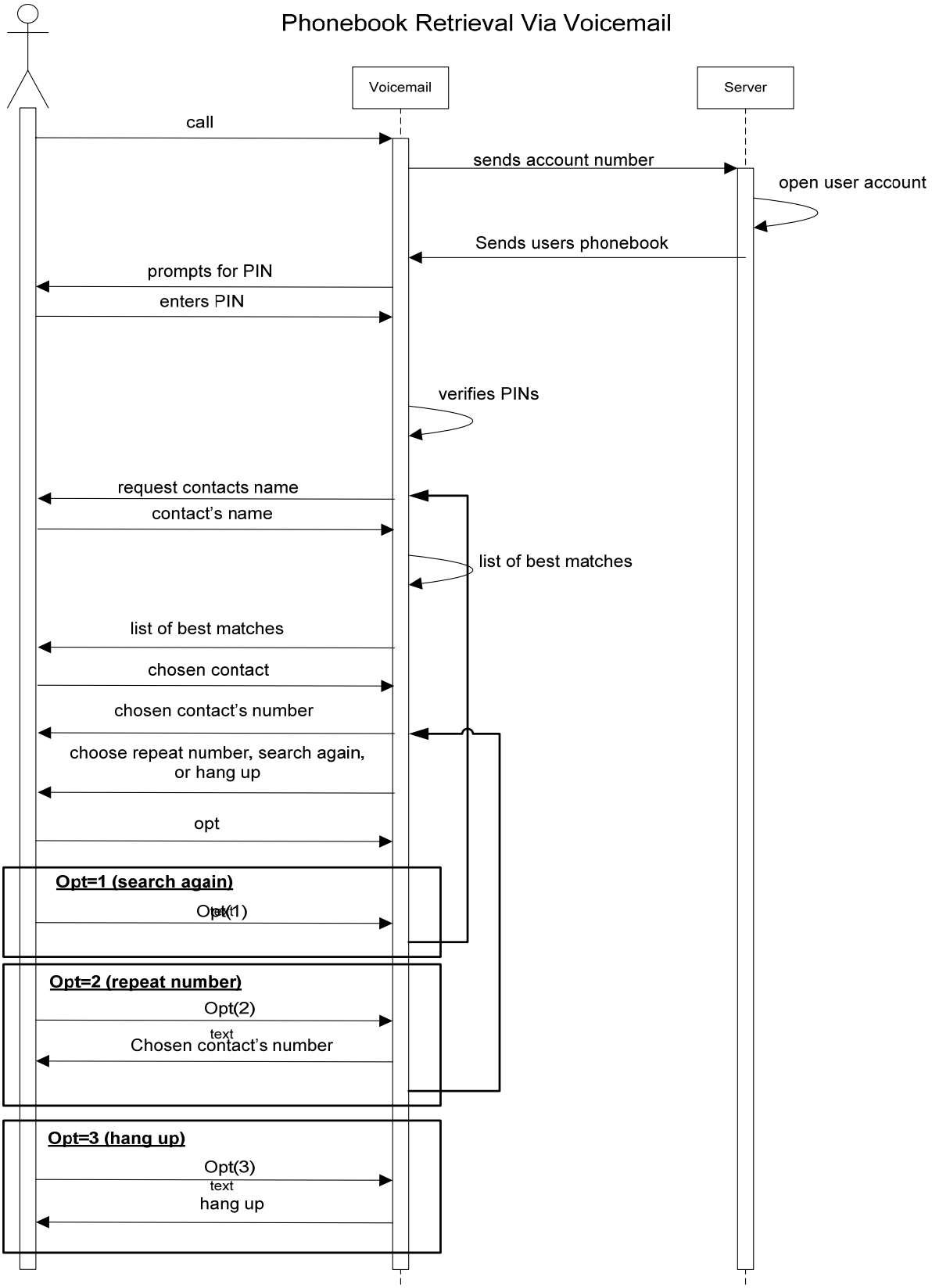


Figure 10.2 Phonebook Access Via Voicemail

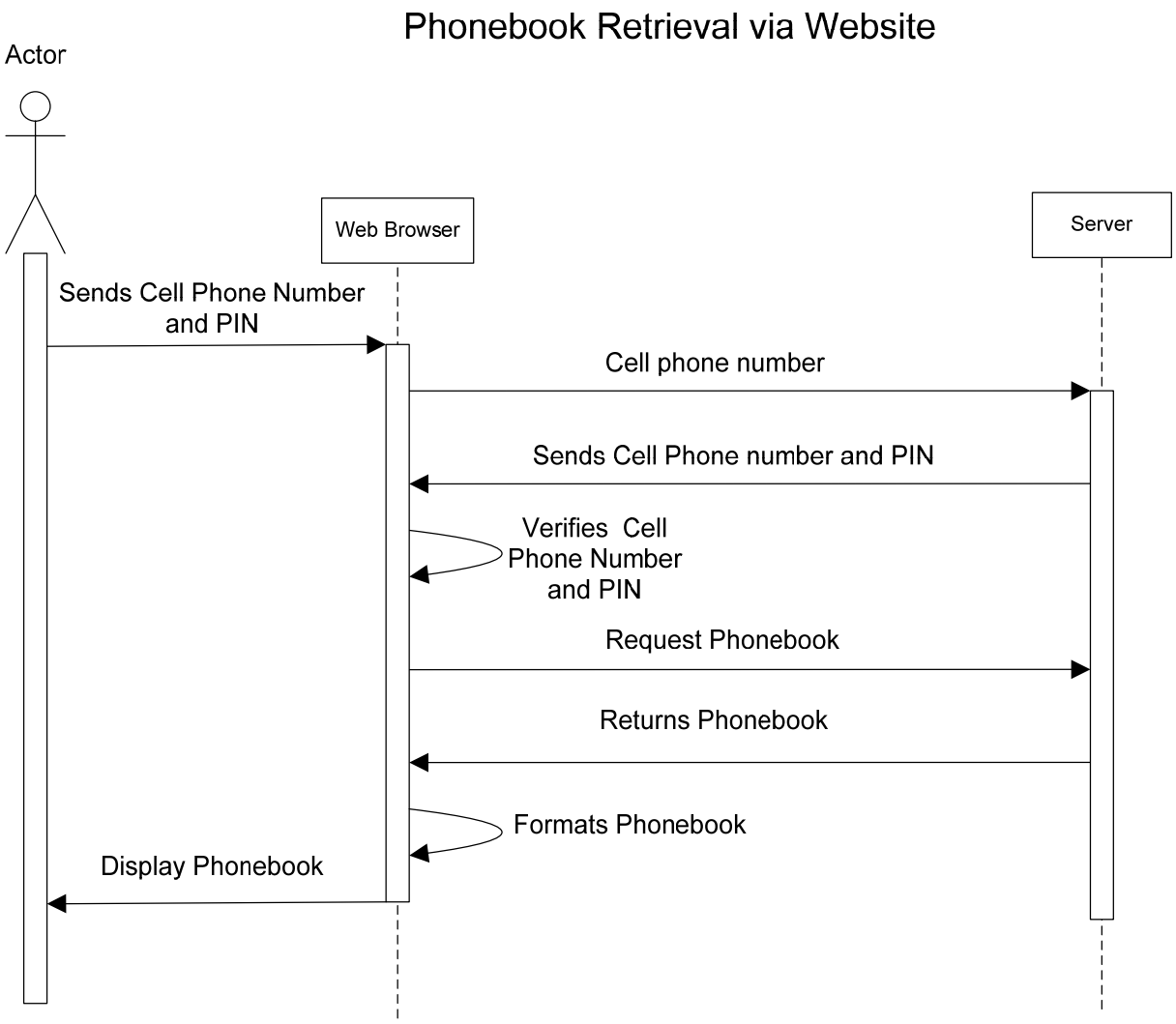


Figure 10.3 Phonebook Retrieval Via Website

11. User Manual

11.1 Introduction

The Centralized Phonebook Database is a system that allows you to access your phonebook from anywhere in the world at anytime.

Previously, if you forgot, broke, or lost your cell phone, you would not be able to retrieve phone numbers stored in the phonebook of your cell phone. The Centralized Phonebook Database (CPD) eliminates this problem while providing a simple and convenient way to access your phone numbers.

The CPD consists of a computer, acting as a server that runs twenty-four hours a day. Your phonebook is stored on the computer to provide access to your phonebook at any time. You may access your phonebook in two different ways:

The first way to access your phonebook is via voicemail access to the CPD. You simply place a call voicemail, use your telephone keypad to enter the name of the person whose phone number you want, and the CPD system will read back the person's phone number. Using voicemail in the CPD system allows you to access your phonebook from any telephone in the world at any time by dialing into your voicemail.

The second method to access the CPD is through a website. Using a web browser, log into the website, enter the name of the person whose phone number you want, and the CPD system returns that person's phone number. The CPD website is helpful for people who prefer a more visual approach to accessing their phonebook.

11.2 Components

There are three main parts to the CPD system: cell phone, voicemail, and website.

1. **Cell Phone:** Most people keep phone numbers in a phonebook stored in their cell phone. A cable and special software to download this phonebook to a computer is available for most cell phones. This allows you to upload the

phonebook into a database maintained by the CPD so that they may be accessed via voicemail from another phone or over the Internet using a common web browser.

2. **Voicemail:** The CPD database may be accessed via any telephone using an interactive voicemail interface. Privacy is protected by requiring the use of a personal identification number (PIN).

3. **Website:** The CPD database may also be accessed through the Internet using a common web browser. The CPD website is password protected, requiring that the user enter the CPD phone number as a username and the voicemail PIN as a password.

11.3 System Requirements

Hardware

- Cell phone with a calling plan from a service provider
- Upload Cable. (Wire to connect your cell phone to your computer.) Can be purchased from your cell phone service provider or cell phone manufacturer. NOT INCLUDED
- Connection to the internet. 56kbps or higher. Speed of phonebook storage to server varies depending on connection speed
- CD-ROM drive
- 10mb storage space on hard drive
- 750mhz or higher computer processor
- 128mb or higher RAM

Software

- Software to store phonebook from cell phone to computer via Upload Cable. This is usually included with your Upload Cable. Note: You MUST be able to save the phonebook in Microsoft Outlook format
- Uploading Software included in CD provided
- Web Browser such as Microsoft Internet Explorer, Mozilla, or Firefox
- Microsoft Windows XP operating system
- Microsoft Outlook 2000 or later version

11.4 Instructions

A third party device must be used to upload your phonebook to your computer prior to use of the Centralized Phonebook Database. We suggest using a DataPilot device (www.datapilot.com). DataPilot makes products which include an upload cable specifically designed for your cell phone and software which extracts your phonebook from you cell phone to your computer. This phonebook must then be saved as a Comma Separated Values (.csv) file. Please see the DataPilot webpage (www.datapilot.com) for more information.

The following are instructions with screen shoots on how to use the Centralized Phonebook Database. The three sections of instruction are for uploading your phonebook, retrieving your phonebook through voicemail, and retrieving your phonebook through the website.

Uploading Phonebook

Getting your phonebook from your computer to the CPD system

Note: Before using CPD, you MUST save your phonebook from your cell phone to your computer by using your Upload Cable and its software. The phonebook MUST be saved in Comma Separated Values (CSV) file format.

1. Turn on your computer and log in as needed.

2. Open your favorite internet browser and type **www.cpd.com** in the address field and push enter. You will see the following screen:

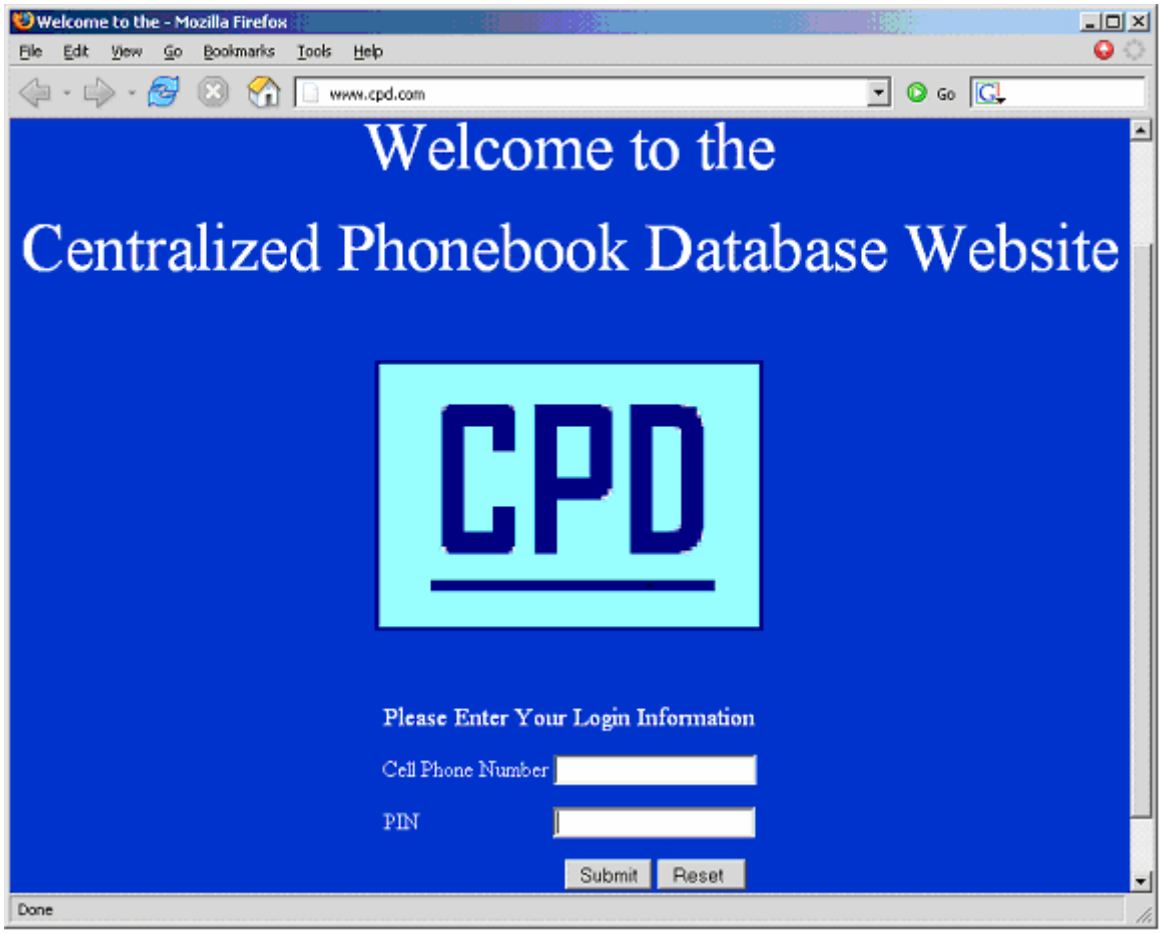


Figure 11.1 Front Page

3. Enter your phone number with area code (without spaces, dashes, or parenthesis) and your voicemail PIN into the labeled text boxes and click on the “Submit” button.

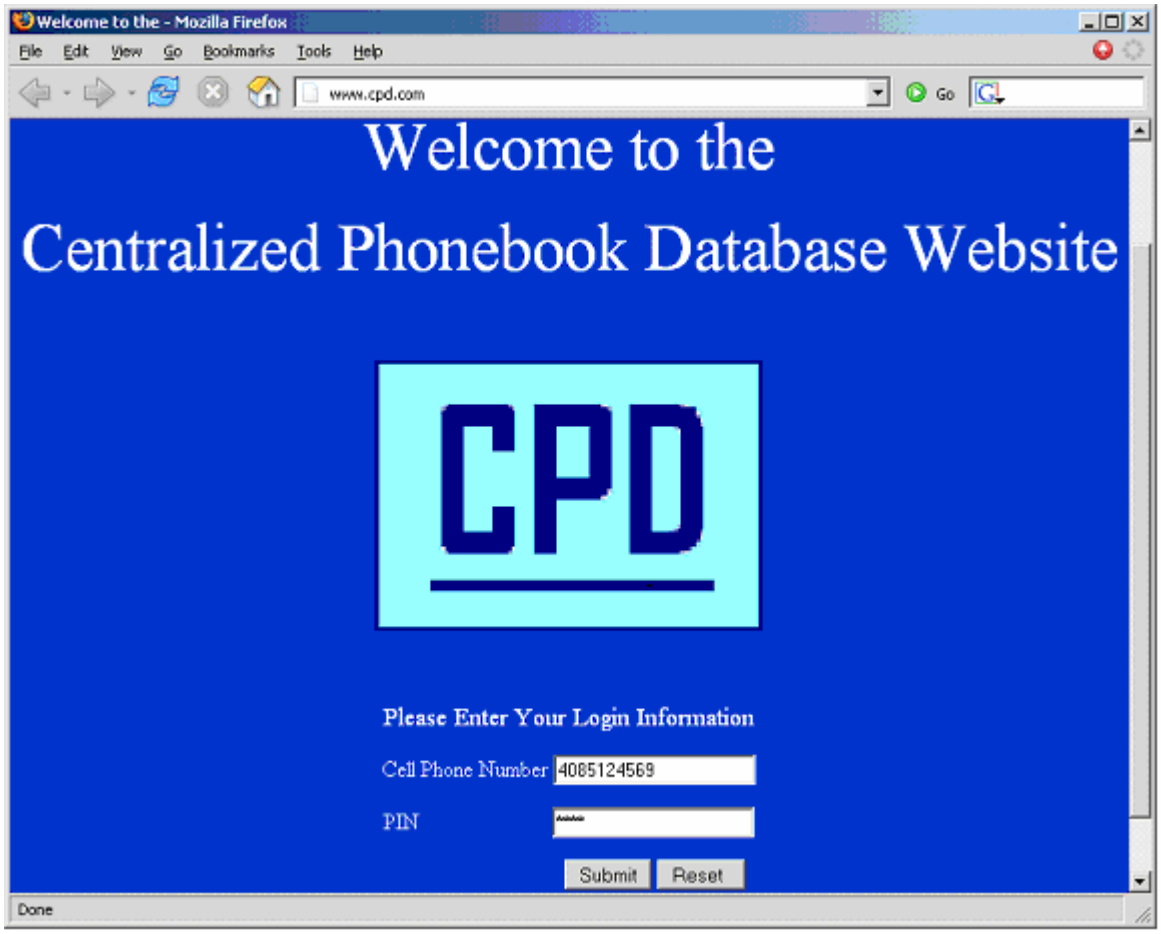


Figure 11.2 Front Page Login

4. Once you are logged in, you will see the following screen. Click on the **Browse** button to find the location of your phonebook.

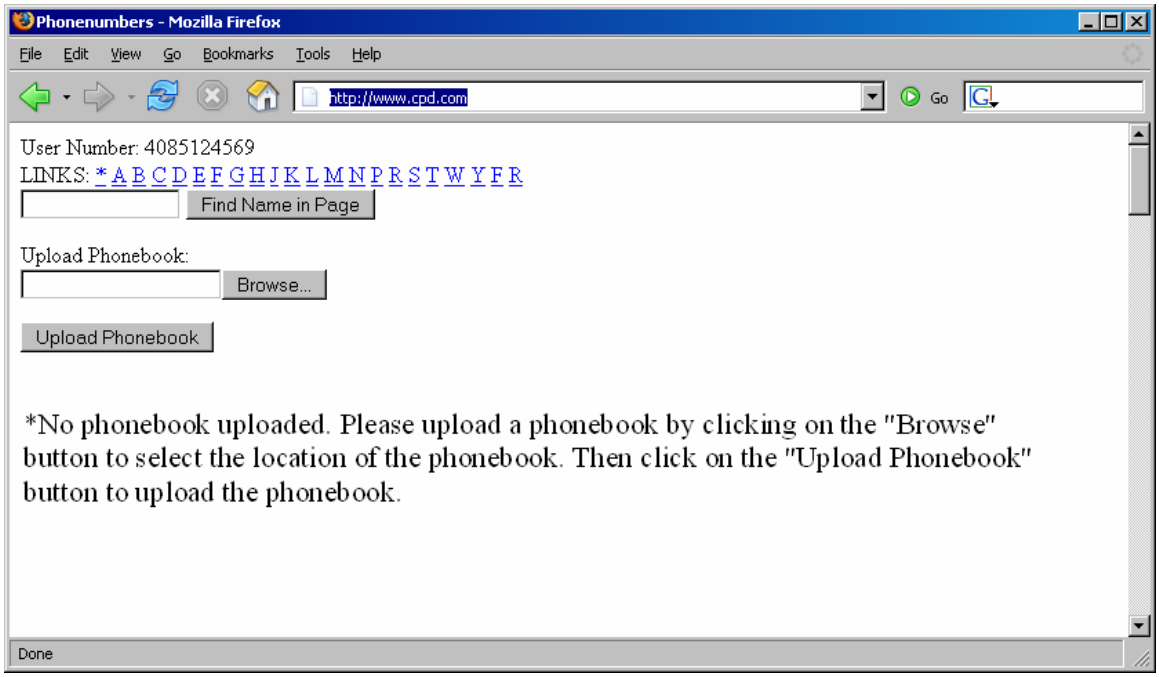


Figure 11.3 Upload Page

5. A Browse window will open. Find the location of your phonebook and click on the **Open** button.

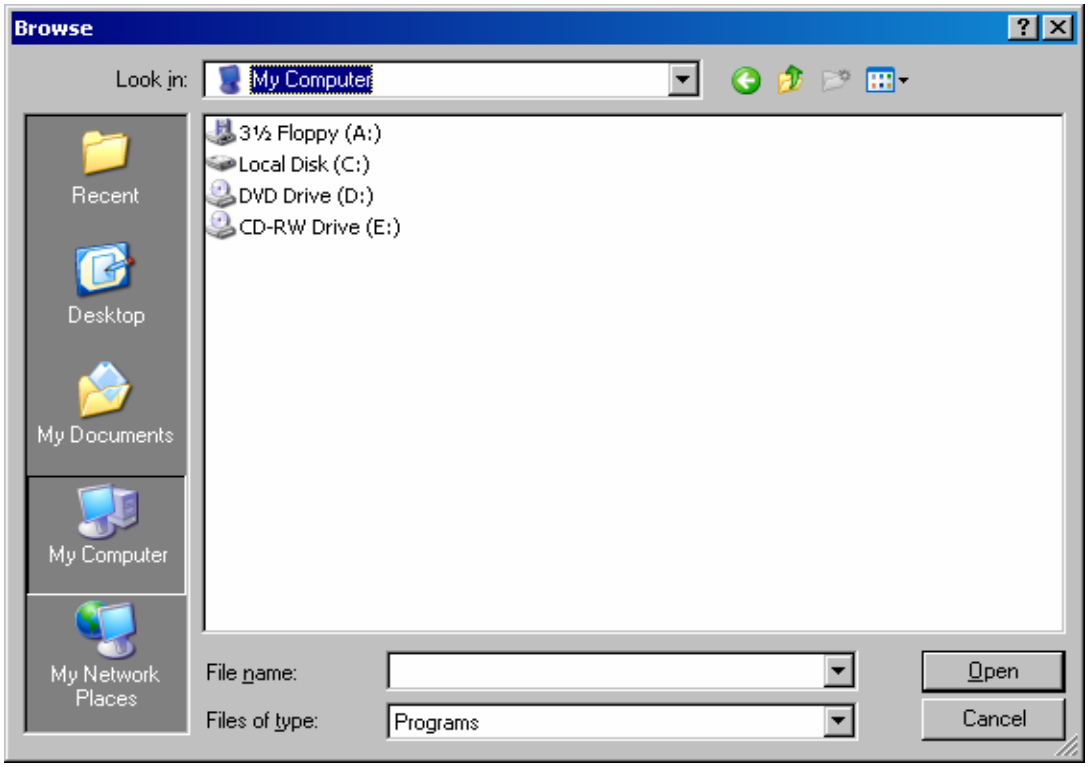


Figure 11.4 File Browse

6. To upload your phonebook, click on the **Upload Phonebook** button.

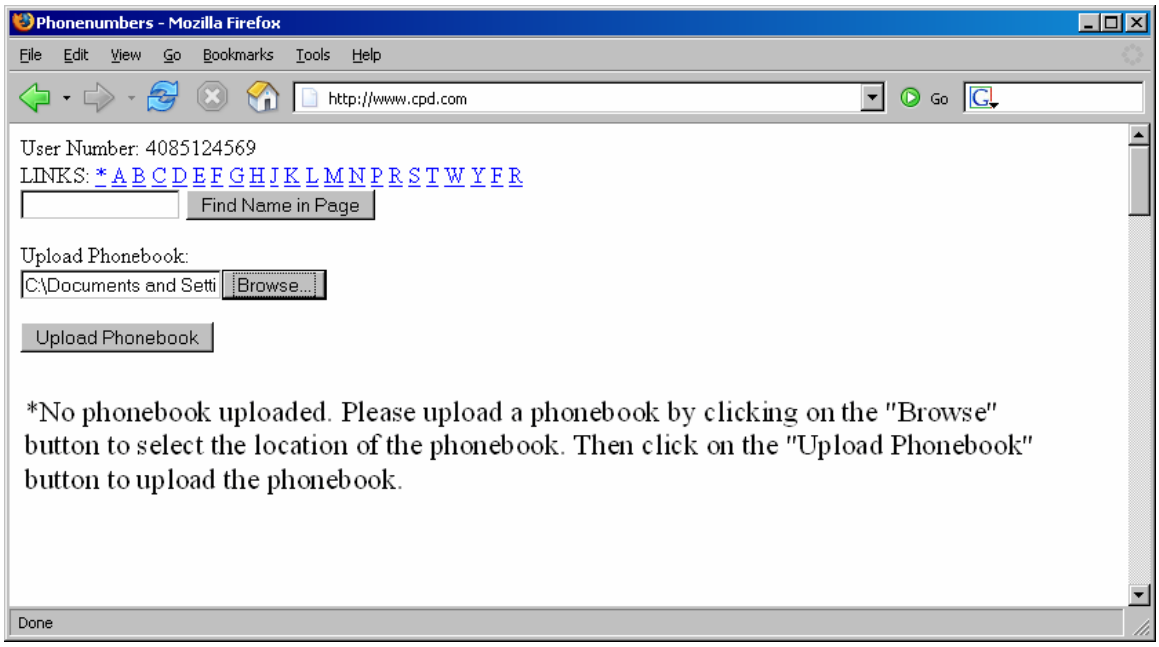


Figure 11.5 Upload Page #2

7. Your phonebook should now be visible. An example is shown below:

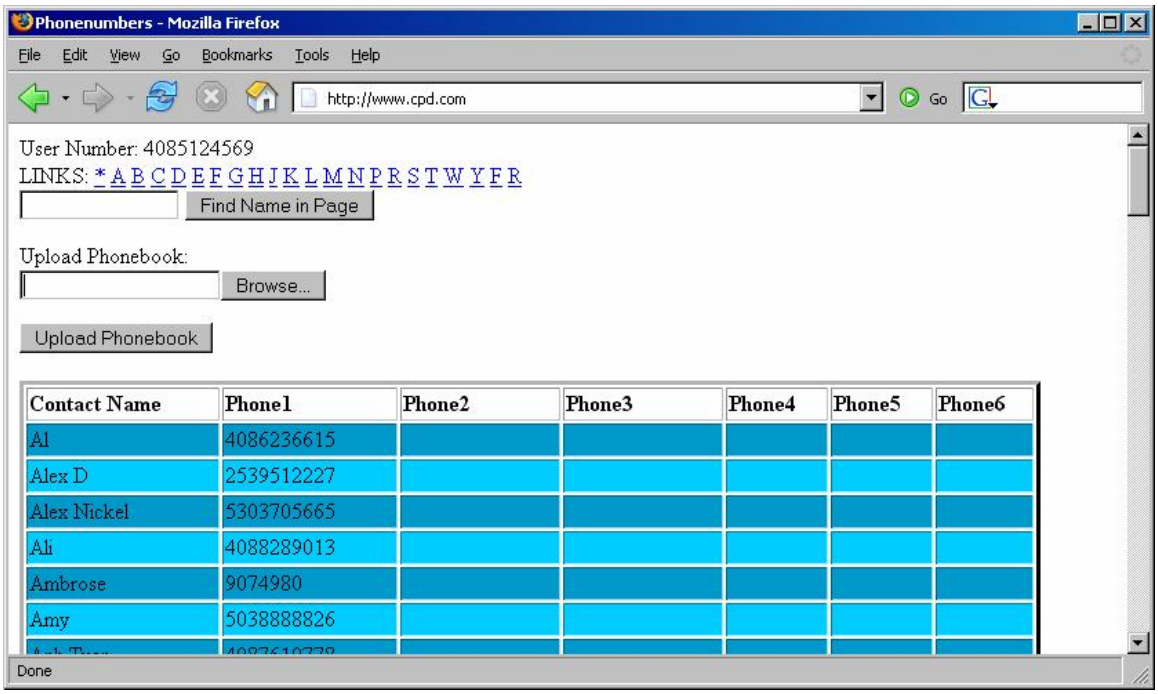


Figure 11.6 Phonebook Display

*The exact number type (home, cell, etc.) of the displayed phone number cannot be displayed due to restrictions in the way DataPilot saves the .csv files. However, there is an overall schema to how the numbers are displayed. If all number types are present, they are presented in the following order: Home1, Home2, Work, Cell, Fax, Pager. If all number types are not entered, the above schema and the ordering can be used to determine which phone number is associated with which number type.

Voicemail Retrieval

Using the Voicemail System:

1. Call your voicemail by dialing your own cell phone number.
2. The voicemail system will ask you for your PIN. Enter your PIN after this prompt.
3. The voicemail system will ask you to enter the name of the person whose phone number you want. Use "T9" text entry format to enter the name and push the pound (#) key when you are done. (See www.t9.com or your cell phone manual if you are not familiar with "T9" text entry format.)
4. The voicemail system will read back a list of possible contacts' names. Enter the number corresponding to the contact's name.
5. The voicemail system will read back the phone number of the name you selected.
6. You will then be given three options. Press 1 to repeat the previous number. Press 2 to enter a new name. This will return you to Step 3 above. Lastly, press 3 or hang up to end the call.

*See flowchart on next page

Voicemail Flow Chart

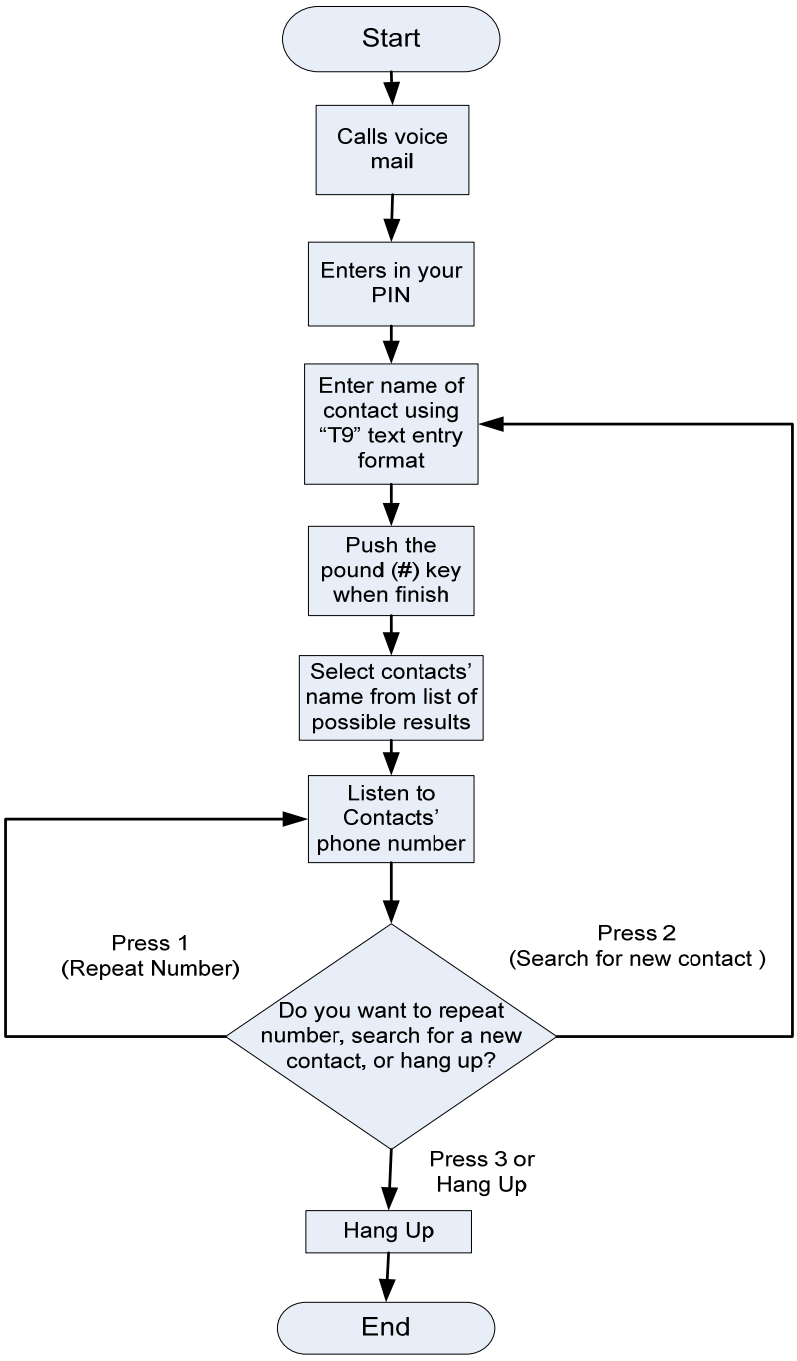


Figure 11.7 Voicemail Flowchart

Web Site Retrieval

Using the website:

1. Open your favorite internet browser and type **www.cpd.com** in the address field and push enter. You will see the following screen:

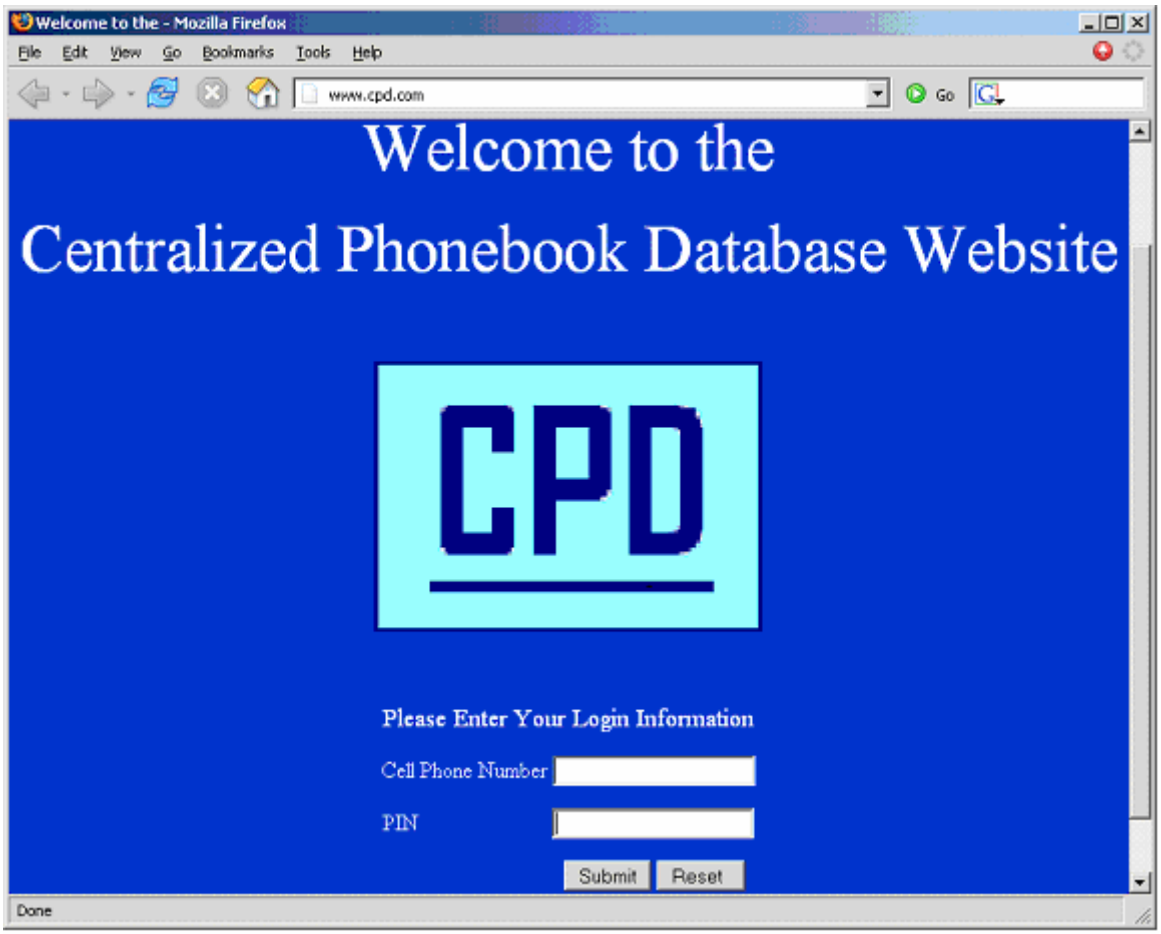


Figure 11.8 Front Page #2

2. Enter your phone number (including area code) and your voicemail PIN into the labeled text boxes and click on the “Submit” button.

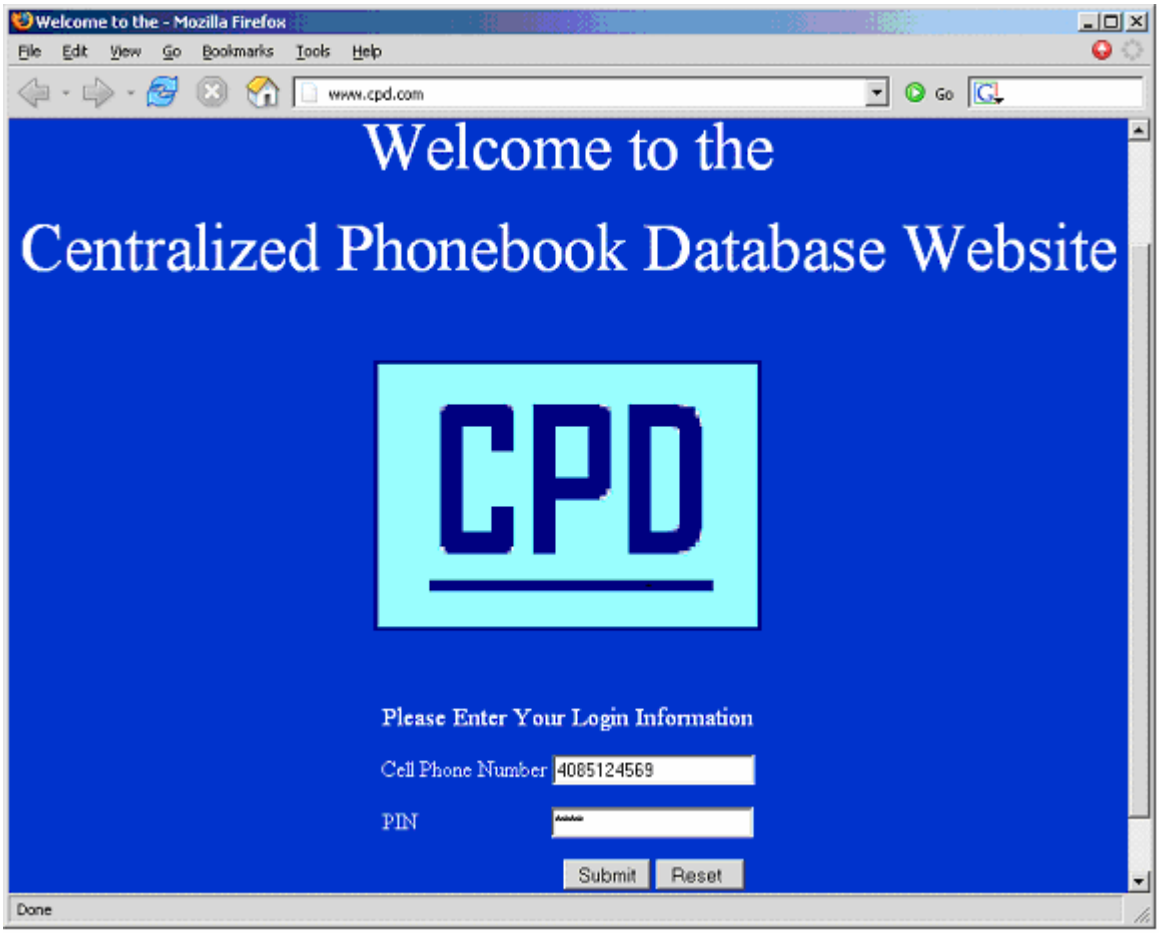
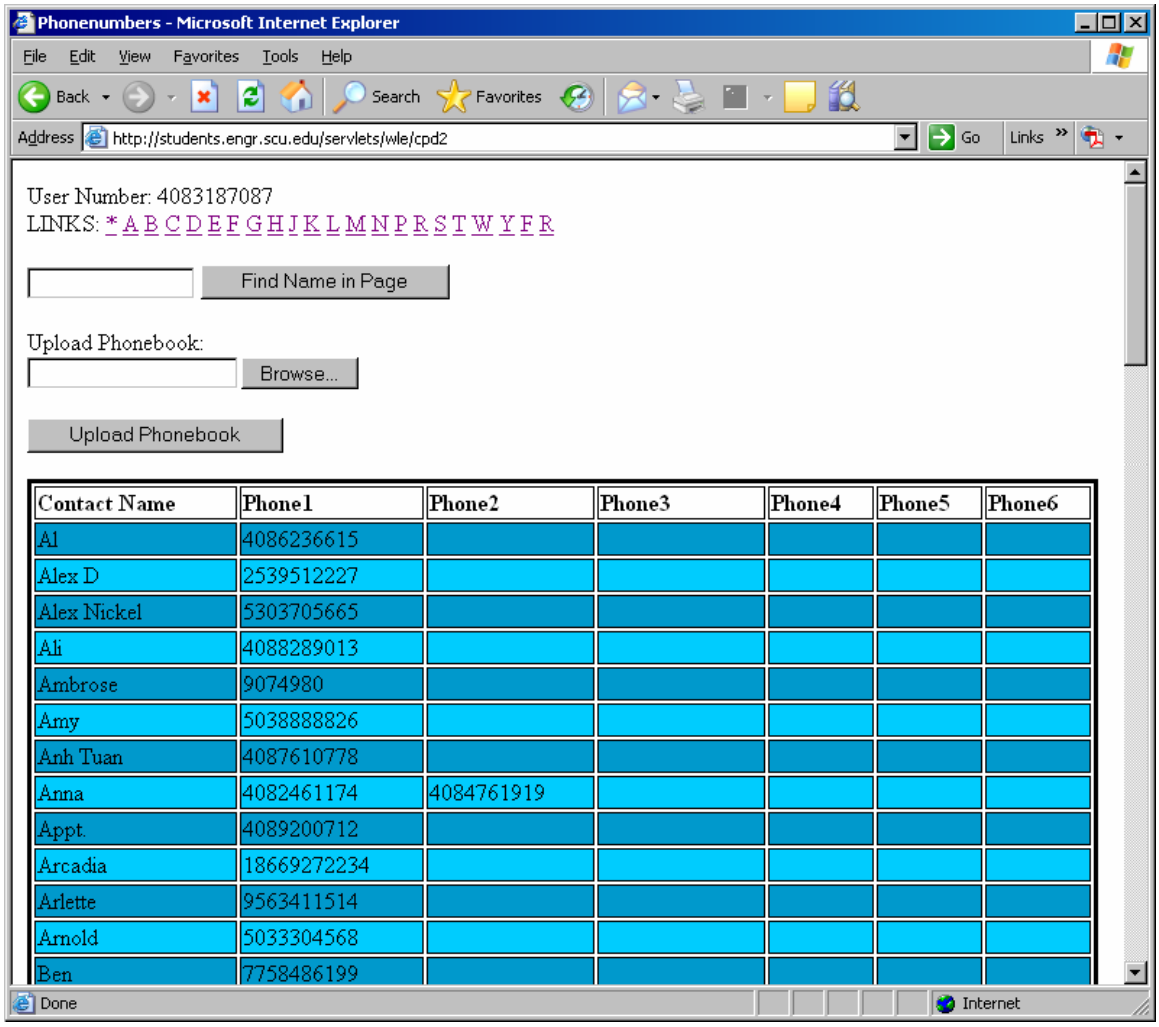


Figure 11.9 Front Page Login #2

3. If you have already uploaded your current phonebook, the following screen will appear, displaying your entire phonebook. You may scroll through the list and find the desired contact or use the search feature. (If you haven't uploaded your phonebook. Please refer to "Uploading Phonebook" section of this manual)



Contact Name	Phone1	Phone2	Phone3	Phone4	Phone5	Phone6
Al	4086236615					
Alex D	2539512227					
Alex Nickel	5303705665					
Ali	4088289013					
Ambrose	9074980					
Amy	5038888826					
Anh Tuan	4087610778					
Anna	4082461174	4084761919				
Appt.	4089200712					
Arcadia	18669272234					
Arlette	9563411514					
Arnold	5033304568					
Ben	7758486199					

Figure 11.10 Phonebook Display #2

4. There are also two ways to search your phonebook. The first way to search is to click on the letter to the right of “LINKS” which corresponds to the first letter of the first name of the contact you are trying to find. This link will scroll the browser to the section that contains names beginning with that letter. (In this example the letter “G” was clicked)

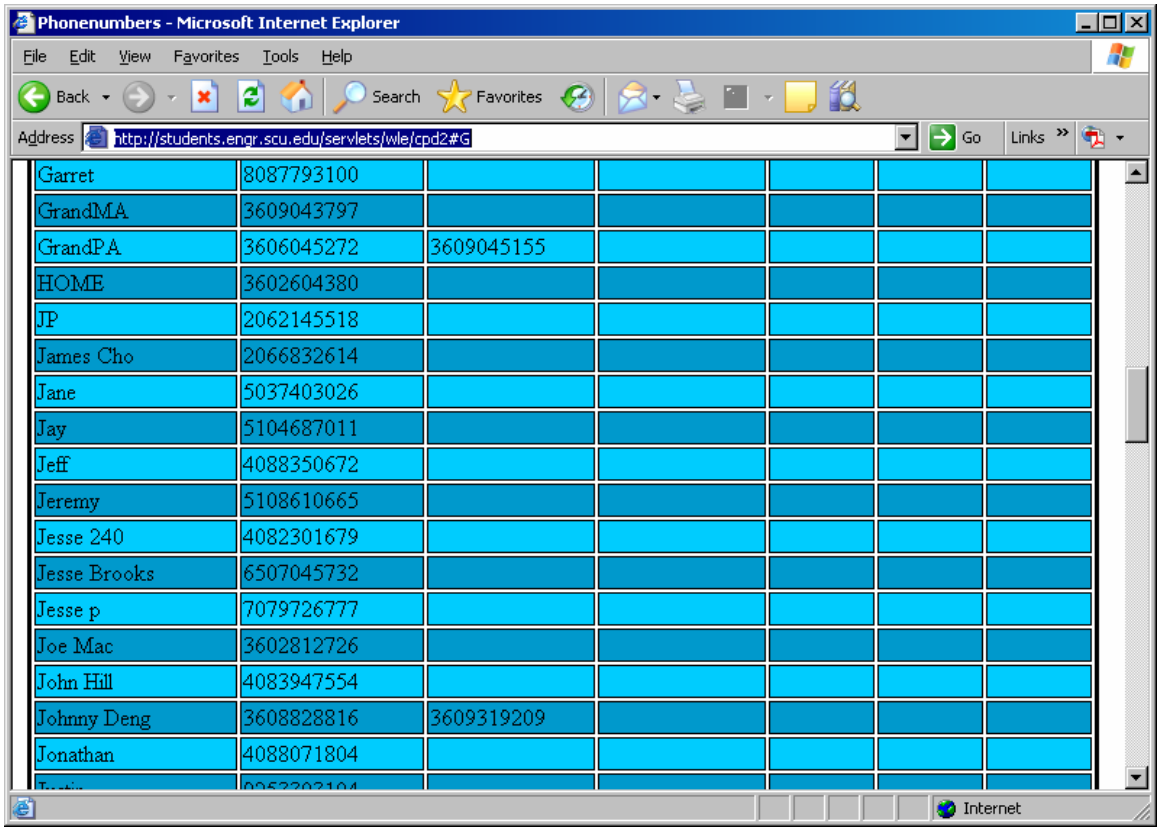


Figure 11.11 Phonebook Link Search

5. The second way to search your phonebook is to type the name of the person whose phone number you want in the search text box and click on the “Find Name in Page” button. The matching contact will be highlighted.

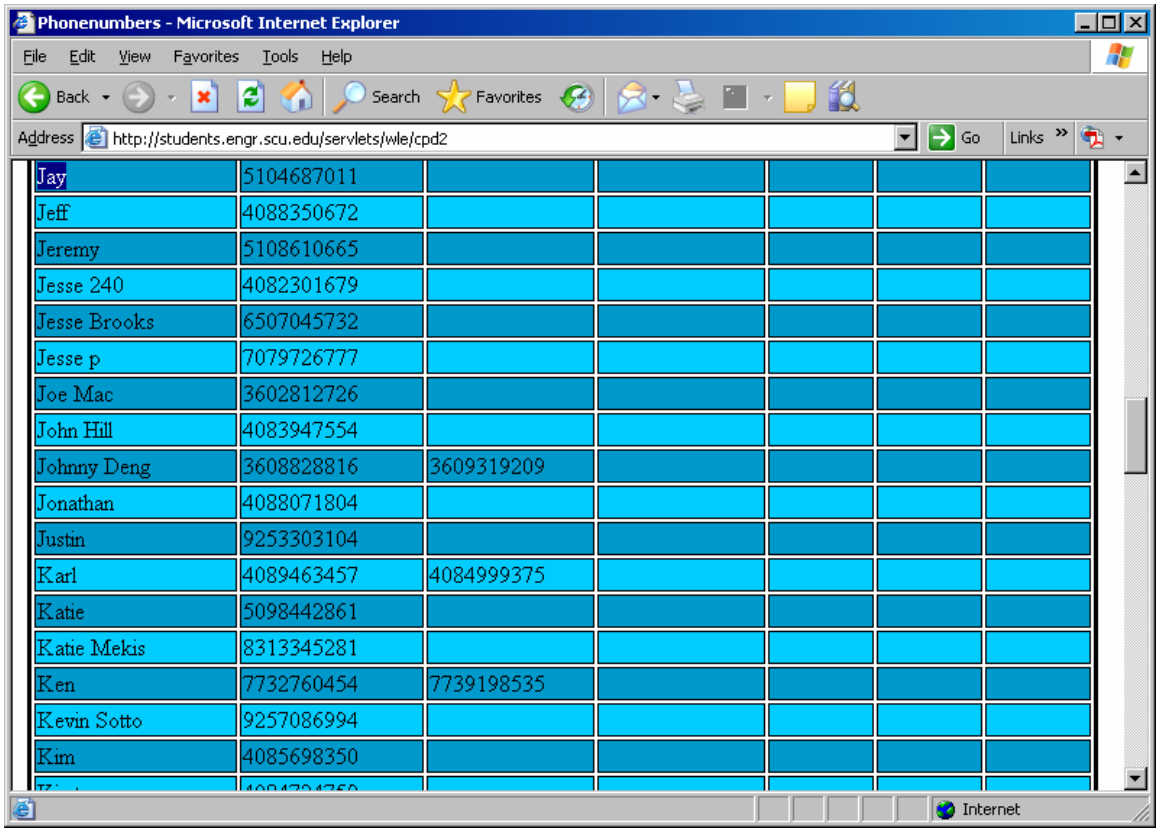


Figure 11.12 Phonebook Typed Search

6. To begin another search scroll back to the top of the webpage and repeat Step 4.

12. Technologies Used

12.1 Software

- DataPilot Universal Kit

  We needed software that would help us extract the phonebook off of a cell phone. Since we did not have access to the cell phones' source code we needed a third party method to achieve this task. We decided on the DataPilot Universal Kit. This kit gave us a simple tool to extract phonebooks from cell phones. In addition, the universal kit was compatible with all but the very newest cell phones, with new phone compatibility always in development. There are also individual DataPilot kits for a person who needs to upload a phonebook from only one phone, which reduces costs.

- Oracle 8i Database

  We needed a way to store all the information that is contained in a phonebook. We decided to use a Database Management System to achieve this task, specifically, Oracle 8i. We chose a Database Management System because it allows for easy creation, modification, and retrieval of data. We chose Oracle 8i because the Santa Clara University Design Center has it installed and its use is free to all engineering students.

- Java Servlets and JDBC

  After deciding to use Oracle 8i we needed a way to display the information on a website. We decided to use Java Servlets and JDBC because they have built in functionality to connect to and interact with databases. In addition, we were able to find a template which made the process even simpler. The Java Servlets run on the web server and take in requests and respond back to the webpage. Java Database Connectivity (JDBC) is called from the servlet and provides a Java Application Programming Interface (API) to connect, send queries to the database, and send information back to the

servlet. The servlet then returns the information back to the website so it can be displayed.

- CGI/Perl

  We needed a way to upload phonebooks to our server. We decided to use Common Gateway Interface (CGI) to achieve this task. CGI allows for an easy way to create an upload field and the back end functionality required to upload a file. We programmed the CGI using Perl, the most common language to implement CGI.

  We also used Perl to manipulate the text contained in the phonebook files (which are stored as Comma Separated Values (CSV) files). We used Perl because one of its best features is text manipulation.

- NCH IVM

  We needed to simulate a voicemail service in order to implement the phonebook retrieval by voicemail. There were a couple of different software options for this task. We decided on the "NCH Swift Sound IVM Phone Answering Attendant Software". This software provided Interactive Voice Response (IVR) capabilities which allowed for number-pad (DTMV) input and voice response. It also let use create a response sequence based on the users' inputs. In addition, the software had modules to connect to a server, download a file, and read from a Comma Separated Values (CSV) file. This was used to download phonebooks in CSV format from the web-server and store it on the local voicemail machine for use by the software.

12.2 Hardware

- MultiTech MT5656ZDX-V modem

  We needed a modem for the voicemail part of the system. The NCH software required that the modem be Telephone Application Programming Interface (TAPI) compliant. The two overall options for this modem were

either getting a voice modem or telephony card. Telephony cards have been shown to have superior functionality but are much more expensive ($300+). On the other hand, voice modems are much cheaper ($50+) but are much less reliable and less likely to work with the software. In the end we decided on a voice modem because of its lower cost. We contacted NCH and asked them for a recommendation on which voice modem to buy. They suggested the MultiTech MT5656ZDX-V voice modem, but they also stated that they do not support or endorse any voice modems. We took their recommendation and bought the modem. At first we have troubles installing the drives and getting all the functionality to work. In the end we were able to get the drives to install correctly and have a complete platform with which to work on.

- DataPilot cable

  We needed a way to upload a phonebook from a phone to a computer. We used the DataPilot cable that came with the DataPilot Universal Kit. The kit provided many cables to connect to different types of phones. The cables had a phone specific connection on one end with a USB connection on the other end to connect to a computer.

13. Project Limitations

- Access to cell phone software

  We did not have access to the source code of the software that is on cell phones. This limited our ability to upload phonebooks directly from a cell phone to the server. We were also unable to download phonebooks to the cell phone from the database and automatically synchronize them (see Possible Enhancements below). Access to the source code would have made for a much more comprehensive and complete system.

- Access to wireless network architecture (including Voicemail)

  We did not have access to telecommunication services and their architecture while creating this system. We had to use personal phone lines and computer voicemail software to implement our design instead of using and editing our real voicemail boxes. This limited the functionality of our voicemail system.

14. Possible Enhancements

- Automatically uploading phonebook straight from phone to server

  Due to our limitations we implemented the phonebook upload through a personal computer. Ideally, the upload would send the phonebook from the cell phone straight to the server with no intermediary steps. In addition, this upload could be automatically uploaded at certain intervals or on command. This would create an always up-to-date phonebook on the retrieval system.

- Phonebook updating on website with synchronization

  An additional feature that could be added is allowing for phonebook editing and updating through the website. This would allow people to change their phonebook on the website in the absence of their phones. Along with phonebook updating could be synchronization, where changes in the online phonebook would be reflected in the phonebook stored in cell phone.

- Phonebook backup and retrieval

  Currently our system stores phonebooks on the server but does not allow for downloading the phonebook back to the phone. An additional feature that could be implemented is to have the phonebooks available for download back to the phone. This would help in situations where phones are permanently damaged or lost and would create a simple way to load a phonebook onto new phone with having to reenter all the names and numbers manually.

- Call forwarding

  Currently the voicemail system reads back phone numbers of desired contacts. An additional feature would be to have the option to automatically forward the call to the desired contact instead of having to write the number down and then call.

15. Societal Issues

A part of being a student at Santa Clara University is understanding how our work affects our society. The following are various societal issues and how our project affects each one.

Ethical: The biggest ethical question with our project is the storage of people's phonebooks and the privacy of the phonebooks and their use. While the system operator may guarantee the confidentiality of each person's phonebook, some people may still be uneasy of having other people store their personal phonebook. These stored phonebooks may also become the desire of governments and if demanded, may have to be handed over to a government agency. These ethical concerns would not arise if the user kept their phonebooks only on their own phone and not in our Centralized Phonebook Database.

Social: This system may have an additional impact on people's memorization of phone numbers, or the lack there of. Most people currently do not memorize phone numbers but this system may increase the number of people who do not memorize phone numbers.

Political: As mentioned in the Ethics section, the information in the Centralized Phonebook Database could come under scrutiny by governments. If the governments wanted the contents of the phonebooks in the system, the system administrator would have to decide weather or not to succumb to government demands.

Economic: The Centralized Phonebook Database would have a minor impact on economics. The phone companies implementing this system would have to put in some development time to create a website, change their voicemail systems, and create the storage for the phonebooks. However, these costs would be very little compared to the benefits achieved by the system. On the user end, users might be charged a nominal fee for the ability to use the system or it may be added as a free feature.

Health and Safety: There are no health or safety issues related to our project. The only possible safety issue is if the wrong person came in possession of phonebooks they were not supposed to have.

Manufacturability: The Centralized Phonebook Database would not be hard to manufacture. It would only take design and implementation time and resources for a complete setup of the system. No new technology or hardware would need to be created which minimizes manufacturing costs.

Sustainability: The Centralized Phonebook Database is very sustainable. Once up and running, it should take little or no managing. The only issue that might arise is the size of the storage used to store the phonebooks. This could be accounted for before creation or dealt with as needed.

Environmental Impact: There should be no environmental impact. The system only adds features to existing architecture and equipment.

Usability: Usability was a major factor during the design of the system. Therefore, there are no major usability issues. The issues are present are due to restrictions, limitations, or lack of a better solution.

Lifelong Learning: This project helped us learn many new skills. The most tangible skill we learned is using new languages and software such as Perl, database interaction, and the voicemail client. More broadly, the project helped us work on a project outside of class with less direction and criteria. This gave us the freedom and difficulty of dealing with a large project.

Compassion: The whole aim of this project is alleviate the suffering of a person who is temporarily separated from their cell phone and needs to contact someone in their phonebook. The project's goal is to offer a service that makes people's lives simpler and easier.

16. Appendix

16.1 References

Html Colors.gif. South Florida Web Design. April 2006. <http://exxia.net/html_colors.gif>.

Instructions for Remotely Logging into the SCU Design Center. Sumit Naiksatam. Santa Clara University. Oct. 2006. <http://students.engr.scu.edu/~snaiksat/ta/ssh.htm>.

MultiTech Systems. 2006. Oct. 2006. <http://www.multitech.com/>.

MultiTech Systems Model MT5656ZDX-Series User Guide. 2004. MultiTech Systems. Oct. 2006. <http://www.multitech.com/DOCUMENTS/Collateral/manuals/S000248G.pdf>.

NCH Swift Sound Software. Nov. 2006. <http://www.nch.com.au/index.html>.

Perl Tutorial. University of Leeds. Feb. 2006. <http://www.comp.leeds.ac.uk/Perl/start.html>.

Reference: HTML Cheatsheet. 2006. Webmonkey. Oct 2006. <http://www.webmonkey.com/webmonkey/index.html>.

Santa Clara University Engineering Design Center Answers to Frequently Asked Questions. 5 June 2001. Santa Clara University. Nov 2006. <http://helpme.scudc.scu.edu/f.a.q.html>.

Sumit Naiksatam's Web Page- Coen 178-Spring 2004. Sumit Naiksatam. 2004. Santa Clara University. Oct. 2006. <http://students.engr.scu.edu/~snaiksat/ta/coen178/spring-2004/>.

The Java Tutorial. 23 Dec. 2005. Sun Microsystems. Oct 2006. <http://java.sun.com/docs/books/tutorial/>.

Transferring Excel and dBASE files to/from Oracle. Feb. 2006. <http://srmwww.gov.bc.ca/gis/dbf2ora.html>.

W3Schools Online Web Tutorials. Refsnes Data. Oct 2006. <http://www.w3schools.com/>.

Wget for Windows. Aug. 2005. April 2006. <http://gnuwin32.sourceforge.net/packages/wget.htm>.

16.2 Source Code

16.2.1 Website Front Page (cpd.html)

```
<HTML>
<HEAD>

<TITLE> CPD Website </TITLE>
</HEAD>

<SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT">
<!--Hide script from older browsers;

  function checkValidation() {
    var idCheck=document.forms[0].elements[0].value;
    var lengthCheck=document.forms[0].elements[1].value;
    var correctId=new RegExp("^[0-9]");
    var correctLength=new RegExp("^[0-9]");

    if ( idCheck == "") {
          alert("Need a valid Cell Phone Number ");
          return false;
    }
    if (!correctId.test(idCheck)) {
          alert("Incorrect Cell Phone Number Format!");
          document.myform.data.select();

          return false;
    }
    if ( lengthCheck == "") {
          alert("Need to enter a PIN!");
          return false;
    }
    if (!correctLength.test(lengthCheck)) {
          alert("Incorrect PIN number format!");
          return false;
    }
    return true;
 }

 // End hiding script -->
</SCRIPT>


</HEAD>

<body bgcolor="#0033CC" text="#FFFFFF">

<FORM method="POST"
action="http://students.engr.scu.edu/servlets/wle/cpd2">
 <!--webbot bot="SaveResults" U-File="/webpages/wle/" S-
Format="TEXT/CSV" S-Label-Fields="TRUE" --><p align="center"> </p>
  <p align="center"> </p>

  <p align="center"><font size="7" color="#FFFFFF">Welcome to the
</font></p>
  <p align="center"><font size="7" color="#FFFFFF">Centralized Phonebook
  Database Website</font></p>
  <p> </p>
  <p align="center">
```

```
  <img border="0" src="Welcom1.gif" width="280" height="196"></p>
  <p> </p>
  <p align="center"><font size="4">Please Enter Your Login
Information</font></p>

  <p align="center">Cell Phone Number <input type="text"
name="cellphonenumber" size="20"></p>
  <p
align="center">PIN         
            
    <input type="password" name="pin" size="20"></p>
  <p
align="center">         &nb
sp;           &nb
sp;           
  <INPUT TYPE="submit" VALUE=" SUBMIT " OnClick="return
checkValidation()" > <INPUT TYPE="reset" VALUE=" RESET
"></FORM><HR>  </button></p>
  <p> </p>



<P>The content of these web pages is not generated by and does not
represent the
views of Santa Clara University or any of its departments or
organizations. </P>
<P class=style3>The content of these web pages is not generated by and
does not
represent the views of Santa Clara University or any of its departments
or
organizations.</P>
<P> </P>
</form>

</body>

</html>
```

16.2.2 Java Servlet and Phonebook Page (cpd2.java)

```
//Programmed By Wilson Le and Anders Hur
//Senior Design 2006
//Code for Website

/**************************************************************************
******
 * CPDSServlet.java

**************************************************************************
*******/

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.PrintWriter;
import java.io.IOException;
import java.sql.*;

/**
 * This class gets data from HTML, writes data to the SQL database,
 * and display results as HTML.
 */
public class cpd2 extends HttpServlet
{
    private String cellphonenumber;
```

```
    private String pin;

    //string to contact SCU server9 with login and password
    String url = "jdbc:oracle:thin:@server9.engr.scu.edu:1521:dc81";
    String login ="wle";
    String passwd = "wle";

    /**
     * This method initialize servlet.
     * @param config The ServletConfig value to be initialized
     * @exception ServletException on servlet error
     * @see ServletException
     */
    public void init(ServletConfig config) throws ServletException
    {
        super.init(config);     //pass servletConfig to parent
    }

    /**
     * This method gets data from database.
     * Displays as an HTML file.
     * @param req HttpServletRequest
     * @param resp HttpServletResponse
     * @exception ServletException on servlet error
     * @see ServletException
     * @exception IOException on input and output error
     * @see IOException
     */
    public void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
        {
            Connection conn = null;
            Statement stat = null;
            ResultSet rs = null;
            StringBuffer sbuf = new StringBuffer();

          // return HTML
            resp.setContentType("text/html");

            // get handle to output stream
            PrintWriter out = resp.getWriter();
            cellphonenumber = req.getParameter("cellphonenumber");
            pin = req.getParameter("pin");

            try {

                DriverManager.registerDriver(new
oracle.jdbc.driver.OracleDriver());
                conn = DriverManager.getConnection(url, login, passwd);
                conn.setAutoCommit(true);
                stat = conn.createStatement();

                //SQL statement
            String qs = "select * from pb,users where pb.usernum= " +
cellphonenumber +" and users.num= " + cellphonenumber + " and users.pin
= " + pin +" order by pb.contactname";

            out.println("<HTML>");
            out.println("<HEAD>");

            //start A-Z links
            out.println("<script>");
            out.println("function init()");
            out.println("     {");
```

```
            out.println("     var tds =
document.getElementsByTagName(\"td\") ;");
            out.println("     var last = '$' ;");
            out.println("     var lnks = \"LINKS: \" ;");
            out.println("     for (i = 7; i < tds.length; i += 7)");
            out.println("           {");
            out.println("           var c1 =
tds[i].firstChild.nodeValue.substr(0,1).toUpperCase() ;");
            out.println("           if (c1 != last)");
            out.println("                 {");
            out.println("                 lnks = lnks + \"<A href=#\" +
c1 + \">\" + c1 + \"</A> \" ;");
            out.println("                 last = c1 ;");
            out.println("                 tds[i].innerHTML = \"<a
name=\" + c1 + \"></a>\" + tds[i].firstChild.nodeValue");
            out.println("                 }");
            out.println("           }");
            out.println("
      document.getElementById(\"ltrs\").innerHTML = lnks ;");
            out.println("     }");
            out.println("</script>");
            //end A-Z links

                out.println("<TITLE>Phonenumbers</TITLE>");
                out.println("</HEAD>");
                out.println("<body onload=\"init()\">");
                out.println("<body>"); //onload=\"init();\">");

            if(qs== null)
            {
                  out.println("<TD> </TD>");
            }
            else
            {
                  rs = stat.executeQuery(qs);
            }

                // Step through the result set
                boolean more = rs.next();
            boolean color= true;


            out.println("<DIV> User Number: " + rs.getString("Usernum")
+ "</DIV>");
            out.println("<DIV id=\"ltrs\">Hello</DIV>");


            //Beginning of Search BOX!!!!!!!!!!!!!!!!!!

            out.println("<script>");
            out.println("<!-- Hide from old browsers");
            out.println("var TRange = null;");
            out.println("var dupeRange = null;");
            out.println("var TestRange = null;");
            out.println("var win = null;");
            out.println("var nom = navigator.appName.toLowerCase();");
            out.println("var agt = navigator.userAgent.toLowerCase();");
            out.println("var is_major   =
parseInt(navigator.appVersion);");
            out.println("var is_minor   =
parseFloat(navigator.appVersion);");
            out.println("var is_ie      = (agt.indexOf(\"msie\") != -
1);");
            out.println("var is_ie4up   = (is_ie && (is_major >= 4));");
```

```
            out.println("var is_not_moz = (agt.indexOf('netscape')!=-
1)");
            out.println("var is_nav     = (nom.indexOf('netscape')!=-
1);");
            out.println("var is_nav4    = (is_nav && (is_major ==
4));");
            out.println("var is_mac     = (agt.indexOf(\"mac\")!=-1);");
            out.println("var is_gecko   = (agt.indexOf('gecko') != -
1);");
            out.println("var is_opera   = (agt.indexOf(\"opera\") != -
1);");
            out.println("var is_rev=0");
            out.println("if (is_gecko) {");
            out.println("temp = agt.split(\"rv:\")");
            out.println("is_rev = parseFloat(temp[1])");
            out.println("}");
            out.println("var frametosearch = self;");
            out.println("function search(whichform, whichframe) {");
            out.println("if (is_ie4up && is_mac) return;");
            out.println("if (is_gecko && (is_rev <1)) return;");
            out.println("if (is_opera) return;");
            out.println("if(whichform.findthis.value!=null &&
whichform.findthis.value!='') {");
            out.println("       str = whichform.findthis.value;");
            out.println("       win = whichframe;");
            out.println("       var frameval=false;");
            out.println("       if(win!=self)");
            out.println("{");
            out.println("       frameval=true;  // this will enable Nav7
to search child frame");
            out.println("       win = parent.frames[whichframe];");
            out.println("}");
            out.println("}");
            out.println("else return;  //  i.e., no search string was
entered");
            out.println("var strFound;");
            out.println("if(is_nav4 && (is_minor < 5)) {");
            out.println("  strFound=win.find(str); // case insensitive,
forward search by default");
            out.println("        }");
            out.println("if (is_gecko && (is_rev >= 1)) {");
            out.println("    if(frameval!=false) win.focus(); // force
search in specified child frame");
            out.println("    strFound=win.find(str, false, false, true,
false, frameval, false);");
            out.println("    if (is_not_moz)
whichform.findthis.focus();");
            out.println("}");
            out.println(" if (is_ie4up) {");
            out.println("  if (TRange!=null) {");
            out.println("
TestRange=win.document.body.createTextRange();");
            out.println("if (dupeRange.inRange(TestRange)) {");
            out.println("TRange.collapse(false);");
            out.println("strFound=TRange.findText(str);");
            out.println("if (strFound) {");
            out.println("        win.document.body.scrollTop =
win.document.body.scrollTop + TRange.offsetTop;");
            out.println("        TRange.select();");
            out.println("        }");
            out.println("   }");
            out.println("   else {");
            out.println("
TRange=win.document.body.createTextRange();");
```

```
            out.println("     TRange.collapse(false);");
            out.println("     strFound=TRange.findText(str);");
            out.println("     if (strFound) {");
            out.println("        win.document.body.scrollTop =
TRange.offsetTop;");
            out.println("        TRange.select();");
            out.println("        }");
            out.println("   }");
            out.println("  }  ");
            out.println("   if (TRange==null || strFound==0) {");
            out.println("
TRange=win.document.body.createTextRange();");
            out.println("   dupeRange = TRange.duplicate();");
            out.println("   strFound=TRange.findText(str);");
            out.println("    if (strFound) {");
            out.println("        win.document.body.scrollTop =
TRange.offsetTop;");
            out.println("        TRange.select();");
            out.println("      }   ");
            out.println("   }");
            out.println(" }");
            out.println("  if (!strFound) alert (\"String '\"+str+\"'
not found!\") // string not found");
            out.println("}");
            out.println("// -->");
            out.println("</script>");
            out.println("<!--  EXAMPLE FORM OF FIND-IN-PAGE SEARCH USING
SUBMIT (ALLOWING 'ENTER/RETURN' KEY PRESS EVENT) -->");
            out.println("<form name=\"form1\"
onSubmit=\"search(document.form1, frametosearch); return false\"><input
type=\"text\" name=\"findthis\" size=\"15\" title=\"Press 'ALT s' after
clicking submit to repeatedly search page\"> <input type=\"submit\"
value=\"Find Name in Page\" ACCESSKEY=\"s\"></form>");

            //END SEARCH BOX


            //BEGIN UPLOAD BOX

            out.println("<FORM ENCTYPE=\"multipart/form-data\"
ACTION=\"/~wle/cgi-bin/upload.pl\" METHOD=\"POST\">");
            out.println("<p>");
            out.println("Upload Phonebook: <BR> ");
            out.println("<INPUT TYPE=\"FILE\" NAME=\"file\">");
            out.println("<p>");
            out.println("<INPUT TYPE=\"submit\" VALUE=\"Upload
Phonebook\">");
            out.println("</FORM>");

            //END UPLOAD BOX


            out.println("<BR><TABLE BORDER=\"3\" BORDERCOLOR=\"BLACK\"
WIDTH=\"740\"> </BR>");
            out.println("<TR>");
            out.println("<TD><B> Contact Name </B></TD>");
            out.println("<TD><B> Phone1 </B></TD>");
            out.println("<TD><B> Phone2 </B></TD>");
            out.println("<TD><B> Phone3 </B></TD>");
            out.println("<TD><B> Phone4 </B></TD>");
            out.println("<TD><B> Phone5 </B></TD>");
            out.println("<TD><B> Phone6 </B></TD>");
            out.println("</TR>");
```

```
            //Outputing the Data in a table
                while (more)
                {
                  //If statement to change the background color of the
table
                  if (color)
                  {
                        out.println("<TR bgcolor=\"#0099CC\">");

                        if ( rs.getString("Contactname") == null )
                              out.println("<TD> </TD>");
                        else
                              out.println("<TD>" +
rs.getString("Contactname") + "</TD>");

                        if ( rs.getString("Phone1") == null )
                              out.println("<TD> </TD>");
                        else
                              out.println("<TD>" +
rs.getString("Phone1") + "</TD>");


                        if ( rs.getString("Phone2") == null )
                              out.println("<TD> </TD>");
                        else
                              out.println("<TD>" +
rs.getString("Phone2") + "</TD>");

                        if ( rs.getString("Phone3") == null )
                              out.println("<TD> </TD>");
                        else
                              out.println("<TD>" +
rs.getString("Phone3") + "</TD>");

                        if ( rs.getString("Phone4") == null )
                                    out.println("<TD> </TD>");
                              else
                                    out.println("<TD>" +
rs.getString("Phone4") + "</TD>");

                        if ( rs.getString("Phone5") == null )
                                    out.println("<TD> </TD>");
                              else
                                    out.println("<TD>" +
rs.getString("Phone5") + "</TD>");

                        if ( rs.getString("Phone6") == null )
                                    out.println("<TD> </TD>");
                              else
                                    out.println("<TD>" +
rs.getString("Phone6") + "</TD>");


                              out.println("</TR>");

                              //using rs.next to get the next value
                              more = rs.next();

                              color =false;
                        }
                        else
                        {
                              //Code to change background color of the
table
```

```
                        out.println("<TR bgcolor=\"#00CCFF\">");

                        if ( rs.getString("Contactname") == null )
                              out.println("<TD> </TD>");
                        else
                              out.println("<TD>" +
rs.getString("Contactname") + "</TD>");

                        if ( rs.getString("Phone1") == null )
                              out.println("<TD> </TD>");
                        else
                              out.println("<TD>" +
rs.getString("Phone1") + "</TD>");

                        if ( rs.getString("Phone2") == null )
                              out.println("<TD> </TD>");
                        else
                              out.println("<TD>" +
rs.getString("Phone2") + "</TD>");

                        if ( rs.getString("Phone3") == null )
                              out.println("<TD> </TD>");
                        else
                              out.println("<TD>" +
rs.getString("Phone3") + "</TD>");

                         if ( rs.getString("Phone4") == null )
                                    out.println("<TD> </TD>");
                              else
                                    out.println("<TD>" +
rs.getString("Phone4") + "</TD>");

                                    if ( rs.getString("Phone5") == null )
                                          out.println("<TD> </TD>");
                                    else
                                          out.println("<TD>" +
rs.getString("Phone5") + "</TD>");

                                    if ( rs.getString("Phone6") == null )
                                          out.println("<TD> </TD>");
                                    else
                                          out.println("<TD>" +
rs.getString("Phone6") + "</TD>");

                                    out.println("</TR>");

                                    //using rs.next to get the next value
                                    more = rs.next();

                                    color =true;
                        }

                  }

                  out.println("</TABLE>");
                  out.println("</BODY>");
                  out.println("</HTML>");
            }

            catch (SQLException e)
            {
                  e.printStackTrace();
                  out.println("<HTML><BODY>");
```

```
                out.println("<BR>SQLException: " + e +
"</BODY></HTML>");
            }

            finally
            {
                try {
                    if (rs != null)
                        rs.close();
                    if (stat != null)
                        stat.close();
                    if (conn != null)
                        conn.close();
                    }
                catch (Exception e2) {
                    e2.printStackTrace();
                }

                // close the output stream.
                out.close();
            }
        }

    /**
     * This method destroys the servlet free resources.
     */
    public void destroy()
    {
        // Servlet is being unloaded, free resources here
    }
}
```

16.2.3 CGI Upload (upload.pl)

```
#!/usr/local/bin/perl

#uploads file to webserver, uploads phonebook to oracle database, and
executes other perl files (functionalities explained below)
 use CGI;
 my $cgi = new CGI;
 my $dir = $cgi->param('dir');
 my $file = $cgi->param('file');
 $file=~m/^.*(\\|\/)(.*)/; # strip the remote path and keep the filename
 my $name = $2;
 open(LOCAL, ">upload.csv") or die $!;
 while(<$file>) {
    print LOCAL $_;
 }

 #redirects webpage
 print "Location: http://students.engr.scu.edu/~wle/test.html\n\n";

 #inserts phone number into beginning of each line in csv file for
upload to database
`perl insertnum.pl upload.csv 4083187087`;

 #parses csv file and uploads to oracle database
 `rsh server9 ". /usr/local/scripts/setup.oracle.sh; cd
/webpages/wle/cgi-bin/; sqlldr wle/wle control=controlfile.ctl skip = 1;
touch ANOTHER.txt"`;

 #removes numbers that were inserted earlier in the csv file
 `perl rmnum.pl upload.csv`;
```

```
 #converts the names of the contacts to numbers so they can be read by
the voicemail software
`perl name2num.pl upload.csv`;
```

16.2.4 Number Insertion (insertnum.pl)

```
#!/usr/local/bin/perl

#inserts phone number into beginning of each line in csv file for upload
to database

if ($#ARGV != 1) {
      print "Usage:  insertnum.pl <csv-file> <phone-number>\n";
      exit;
}

$number=$ARGV[1];
$csvfile=$ARGV[0];

open(FILE, "$csvfile") or die "Could not open file\n\n";

@lines = <FILE>;

#inserts number and quotations infront of each line
for $lines (@lines)
{
      substr($lines, 0, 0) = $number . ",";
      $lines =~ s/"//g;
      #@split = split(/,/, $lines);
      #$split[3] = "\u$split[3]");
      #$lines = join(",",@split);
}

close FILE;

open(FILE, ">$csvfile") or die "Could not open file\n\n";

for $lines (@lines)
{
      print FILE $lines;
}


close FILE;
```

16.2.5 sqlldr Control File (controlfile.ctl)

```
load data
infile upload.csv
replace
into table pb
fields terminated by ','
(usernum, entryNum FILLER, contactName, phone1 , phone2, phone3, phone4,
phone5, phone6, email1 FILLER , email2 FILLER , email3 FILLER , street
FILLER, postbox FILLER, city FILLER, state FILLER, zip FILLER, country
FILLER, memo FILLER)
```

16.2.6 Number Removal (rmnum.pl)

```
#!/usr/local/bin/perl

#removes numbers that were inserted earlier in the csv file
```

```
if ($#ARGV != 0){
      print "Usage:  rmnum.pl <csv-file>\n";
      exit;
}

$csvfile=$ARGV[0];

open(FILE, "$csvfile") or die "Could not open file\n\n";

@lines = <FILE>;

#removes the previously inserted number and quotations from the
beginning of each line
for $lines (@lines)
{
      substr($lines, 0, 11) = "";
}

close FILE;

open(FILE, ">$csvfile") or die "Could not open file\n\n";

for $lines (@lines)
{
      print FILE $lines;
}

close FILE;
```

16.2.7 Convert Name to Numbers (name2num.pl)

```
#!/usr/local/bin/perl

#converts the names of the contacts to numbers so they can be read by
the voicemail software

if ($#ARGV != 0){
      print "Usage:  name2num.pl <csv-file>\n";
      exit;
}

$csvfile=$ARGV[0];

open(FILE, "$csvfile") or die "Could not open file\n\n";

@lines = <FILE>;

close FILE;

#z used for index of new array with new file contents
$z = 0;

open(NEWFILE, ">upload.csv") or die "Could not open file\n\m";

foreach $line (@lines)
{
      reset '$newLine';

      #print "\n\n j = $j \n\n";
      $line =~ s/"//g;
      print $line;

@split = split(/,/, $line);
```

```
$lengthQ = length($split[1]);

$onlyName = substr($split[1], 0, $lengthQ);

$length = length($onlyName);

#converts letters to numbers (as is on phone number-pad)
for( $i = 0; $i < $length; $i++)
{
      $char = substr($onlyName, $i, 1);

      if (($char eq A) || ($char eq a) || ($char eq B) || ($char eq b)
|| ($char eq C) || ($char eq c))
      {
            $char = 2;
      }

      elsif (($char eq D) || ($char eq d) || ($char eq E) || ($char eq
e) || ($char eq F) || ($char eq f))
      {
            $char = 3;
      }

      elsif (($char eq G) || ($char eq g) || ($char eq H) || ($char eq
h) || ($char eq I) || ($char eq i))
      {
            $char = 4;
      }

      elsif (($char eq J) || ($char eq j) || ($char eq K) || ($char eq
k) || ($char eq L) || ($char eq l))
      {
            $char = 5;
      }

      elsif (($char eq M) || ($char eq 'm') || ($char eq N) || ($char eq
n) || ($char eq O) || ($char eq o ))
      {
            $char = 6;
      }

      elsif(($char eq P ) || ($char eq p ) || ($char eq Q ) || ($char eq
q ) || ( $char eq 'R' ) || ($char eq r )
|| ($char eq S ) || ($char eq 's' )))
      {
            $char = 7;
      }

      elsif(($char eq T ) || ($char eq t ) || ($char eq U ) || ($char eq
u ) || ($char eq V ) || ($char eq v ))
      {
            $char = 8;
      }

      elsif(($char eq W) || ($char eq w) || ($char eq X) || ($char eq x)
|| ($char eq Y) || ($char eq 'y') || ($char eq Z) || ($char eq z))
      {
            $char = 9;
      }

      else
      {
            $char = '';
```

```
	}

	$numName = $numName . $char;
}

#insert dashes into phone numbers
for($a = 2; $a < 17; $a++)
{
	if (length($split[$a]) == 10)
		{
			#$split[3] = substr( $split[3], 0, 3);
			$split[$a] = (substr( $split[$a], 0, 3) . "-" .
substr( $split[$a], 3, 3) . "-" . substr( $split[$a],
6));
		}

	elsif(length($split[$a]) == 11)
		{
			$split[$a] = ( substr( $split[$a], 0, 1) . "-" .
substr( $split[$a], 2, 3) . "-" . substr(
$split[$a], 4, 3) . "-" . substr( $split[$a], 7));
		}
}

$split[1] = $numName;

$newLine = join(",",@split);

print $newLine . "\n";
	$dumline =  $newline;

	print NEWFILE $newLine;
}

close NEWFILE;
```