Meeting Scheduler Using Mobile Agents

Robert K. Clark III
University of Illinois at Springfield
Springfield, Illinois 62703
bobbyalicia@insightbb.com

Abstract

Microsoft Outlook gives users a great tool to organize hectic schedules and provides the ability to be reminded of tasks for those who are forgetful. One of the shortfalls of Microsoft Outlook is that there has to be some kind of server (i.e. Microsoft Exchange Server) in order to connect all of the users. If all of the users are not connected, the schedules of each user cannot be shared amongst users, managers, supervisors, and others throughout the company. Companies with 100 or fewer employees would benefit from small and agile software that could allow supervisors to schedule meetings for employees without the need for a server or central processing machine to connect the users. The software should be small and agile without the need for a large number of resources. This Meeting Scheduler can be written using small and agile pieces of software called mobile agents.

1 Rationale and Significance

Inside companies throughout the world, managers and supervisors gather employees for meetings multiple times a month. Software that can provide information about employees' appointments and schedules is very important because it helps managers and supervisors know when and when not to schedule meetings. Microsoft Outlook is a widely used software program that allows users to schedule personal tasks and make appointments. Microsoft Outlook gives users a great tool to organize hectic schedules and provides the ability to be reminded of tasks for those who are forgetful. One of the shortfalls of Microsoft Outlook as a stand-alone program in an office environment is that there has to be some kind of server (i.e. Microsoft Exchange Server) in order to connect all of the users together. If all of the users are not connected, the schedules of each user cannot be shared amongst users, managers, supervisors, and etc throughout the company. While many large companies and corporations probably have this server or similar software, some smaller companies with 100 or fewer employees may not find it necessary to purchase software of that magnitude. Companies in this situation would benefit from software that could aid in scheduling meetings for employees without the need for a server or central processing machine to connect the users. The software should be small and agile without the need for a large number of resources.

2 Problem Statement

The author proposes to develop software that allows a user to retrieve the schedules of designated users and use that data to determine the best time to schedule a meeting. This software would use mobile agents to go across the network to retrieve data from each user's computer. The agent will retrieve appointments from Microsoft Outlook and send the dates and times of those appointments or meetings to the user that requested it. The data will then be analyzed to determine a common time when all requested employees are available. The mobile agents will be written in Java using the Aglets Work Bench developed by International Business Machines (IBM). The software will be able to be used by any user in the company. The goal of this software is to give managers, supervisors and other administrators the ability to schedule meetings without the need for software (such as Microsoft Exchange Server) that connects all users' calendars together.

3 Literature Review

3.1 What about other company's software?

Other companies have made products similar to the one that I'm proposing. In an attempt to distinguish my application from others, I searched the Internet for similar meeting scheduler. The author researched three separate meeting scheduler applications. Two trials were downloaded out of those three. The first downloaded product was called Meeting Maker. It is meeting scheduler software that uses its own storage for meetings and calendar information. This software does not use the calendar storage facilities of Microsoft Outlook. The next downloaded application was Meeting2000. This application uses Microsoft Outlook to store appointments and meetings. It also synchronizes meetings with Microsoft Outlook. The thing that it doesn't do is look at what times individuals and conference rooms are available. It relies on the user to know when employees and conference rooms are available. The last product I found was Share 360 Scheduler. It was a software package that required a server, a web server and uses a web browser interface to access the application. This application had a complicated setup and required many other pieces of software in order to properly operate. This application required more setup and maintenance than the application that will be developed with this project. Many products integrate Microsoft Outlook, but those products still rely on Microsoft Outlook to determine what dates and times users are available for meetings. Managers and supervisors still cannot look at what dates and times employees are available because they don't have software that goes and looks at employee calendars. This is where my meeting scheduler will be superior.

3.2 What are mobile agents?

Mobile agents are independent pieces of software that can be dispatched from one computer to another computer to perform a task and then return with results. Mobile agents are usually small, agile, flexible, and collaborate with some sort of host (Wang 2). There are many different kinds or flavors of agent technology. Examples of mobile agent technology are Telescript (Cockayne 33), IBM Aglets Workbench and Agents for Remote Access (Ara) (Cockayne 33). The mobile agent technology that will be used for this meeting scheduler application is IBM Aglets Workbench (Cockayne 33).

3.3 Why use mobile agents?

Mobile agents solve the well-known problem that plagues client/server applications. Each request or query that a client performs has to make trips over the network. These queries consume network bandwidth and as more and more trips are made, performance suffers. Instead of having multiple transactions that travel over the network, one mobile agent can be sent over the network to perform the transactions and return when the transaction is completed (Sundsted 1). Another problem that client/server applications encounter is that operations between a client and a server stop if the network connection is lost. An agent can be setup so that the client that sent the agent can go offline while the agent is running elsewhere on the network. The agent will then wait for the client to become online again before transmitting the results. Client/server transactions normally have to restart if this were to happen (Wang 2). Mobile agents are also great at adapting to the environment in which they are running and they also become an independent process that does not immediately rely on the process that dispatched it (Lange 88).

3.4 Why use the IBM Aglets Workbench?

The IBM Aglets Workbench was chosen because of its foundation. It was built upon the Java programming language that has some of the best facilities for network programming. The aglets workbench allows the developer to create Java applications that dispatch aglets to remote computers to perform the preprogrammed tasks. Another reason the aglets workbench was selected was the fact that Java is platform independent and would allow the application on many different platforms (Cockayne 166). The final reason (the most important reason) is the fact that Java has a great security layer. The Java security framework allows the implementation of digital signatures and encryption as well as a few other secure methods if needed (Cockayne 170).

3.4.1 Aglet security

Aglet security is of the utmost importance when dealing with mobile agents that will move from computer to computer and access local resources. Companies normally setup firewalls that allow employees to access the Internet, but rarely allow computers outside the firewall to request information from computers behind the firewall. Secure information would have to be encrypted before it was sent over the network otherwise, because some third party software could intercept the data and read its contents (Kun 22). Part of the problem with firewalls is that it assumes that all applications that are executing are due to the requests of users behind the firewall. That assumption couldn't be farther from the truth. When users open email attachments, those attachments may include

executable code that performs some set of instructions that may leave the computer vulnerable to attack. Sometimes this mobile code could become a mobile agent by moving itself inside the network. Anything that happens after this point would appear as an internal attack because the code is being run from a machine on the network (Kun 23). Several steps can be used to ensure safe operation of mobile agents. These steps include authenticating the mobile agent, verifying its code and restrictions to local resource access (Kun 24). Security is also ensured because each computer an aglet visits must run an "aglet host". This aglet host provides an environment for an aglet to operate just as a web browser gives an applet an environment to execute. This is still one of the main limitations of aglets. The aglet host is started using local security policies to prevent the running of malicious code.

3.4.2 Uses for Aglets

Aglets have many uses in the mobile computing world. Aglets have a wide range of uses in different industries. A purchasing aglet could help a consumer compare prices for an item among competitors. Individuals involved in finance could have an aglet that keeps track of stock prices and could notify a user if a particular stock reaches a certain price. The same concept applies to those in the travel industry. Travel agents could use aglets to go fetch the best prices for hotels, car rentals and airfares (Cockayne 11).

4 Plan of Work

4.1 Analysis of Requirements

A list of features that were included in the meeting scheduler software is shown in Table 1. The software was written in an object-oriented manner and was mostly written using the Java programming language. A GUI interface is provided to allow user-friendly operation of the meeting scheduler software. A list of hardware and system requirements for the software is included in this report. A readme document, on screen help and Installshield installation were provided to help users setup the aglet host and application.

Feature	Support
Check the availability of users	Yes
Email notification to attendees of a meeting	Yes
Schedule meeting in Outlook	Yes
Help and Setup Documentation	Yes
Graphical User Interface	Yes

Table 1: Supported Features

4.2 Design

The design of the meeting scheduler software had two separate areas of work. The first area dealt with dispersing the mobile agents to the specified machines to retrieve appointment data from a user's calendar that is located in Microsoft Outlook and display that data to the meeting scheduler. This area required the careful development of dynamic link libraries that are used by the meeting scheduler to retrieve appointment information using the Windows API. Care was taken when parameters were passed across different programming languages. One dynamic link library (or DLL) will be required to be installed on each user's computer to facilitate the Windows API calls that were used to retrieve appointment information. The last area dealt with using a mobile agent to create appointments on the attendee's machine and allow the meeting scheduler user to send notices about a meeting to users. The users responded to a question to make sure the appointment was okay to make and if they could accept the meeting request, the appointment was made on the user's computer. This application only works in Microsoft Outlook 2000, Outlook XP and Microsoft Outlook 2003. Help is built into the application and an Installshield installaton is also provided to assist the user in installing the mobile agent service that will be required to allow mobile agents to run on the machine if time allows.

The Meeting Scheduler dispatched mobile agents to each user's computer and retrieves dates and times of current appointments from the Microsoft Outlook calendar on the local computer. It returned to the computer that dispatched it with a list of current appointments for each user. The only item that needed to be known is the IP address or computer name of each person's computer that needs to be included in the scheduling process. This is how the Meeting Scheduler knew where to go to find calendar information. The person scheduling the meeting had an option to specify priority for times. These options included a specific day or a specific time of day. The available time frame that most closely matches the highest priority option was used to create the appointment. After a meeting time was decided by the software, a message was sent to each person that will attend the meeting to let the individual know that a meeting is being scheduled. Each user had to agree to the meeting time by

confirming the appointment. This was accomplished by displaying a pop-up message to the user asking for approval to make the appointment on the calendar. The appointment won't be made on the user's calendar until they confirm the date and time. If any user declines, the meeting scheduler will notify the person that is scheduling the meeting. Confirmation of the appointment was achieved through two different methods. A pop-up window or an email was the two methods used to perform confirmation. If pop-up was selected, the user received a pop-up message telling them that an appointment was scheduled on their Outlook calendar. If email was selected, the user received an email telling them that an appointment had been scheduled on their Outlook calendar. Both options could be selected at the same time.

4.3 Implementation

The software implementation consisted of completing the data retrieval area, then the dispersion of the agents and getting the data back to the user and finally analyzing the data and developing the user interface. Since each section of the software is independent of the other sections, any design changes to a section had little or no bearing on the other sections of the software that were being developed. This assisted in the implementation process because the entire product did not have to be redesigned or overhauled because of a design flaw in one of the other sections.

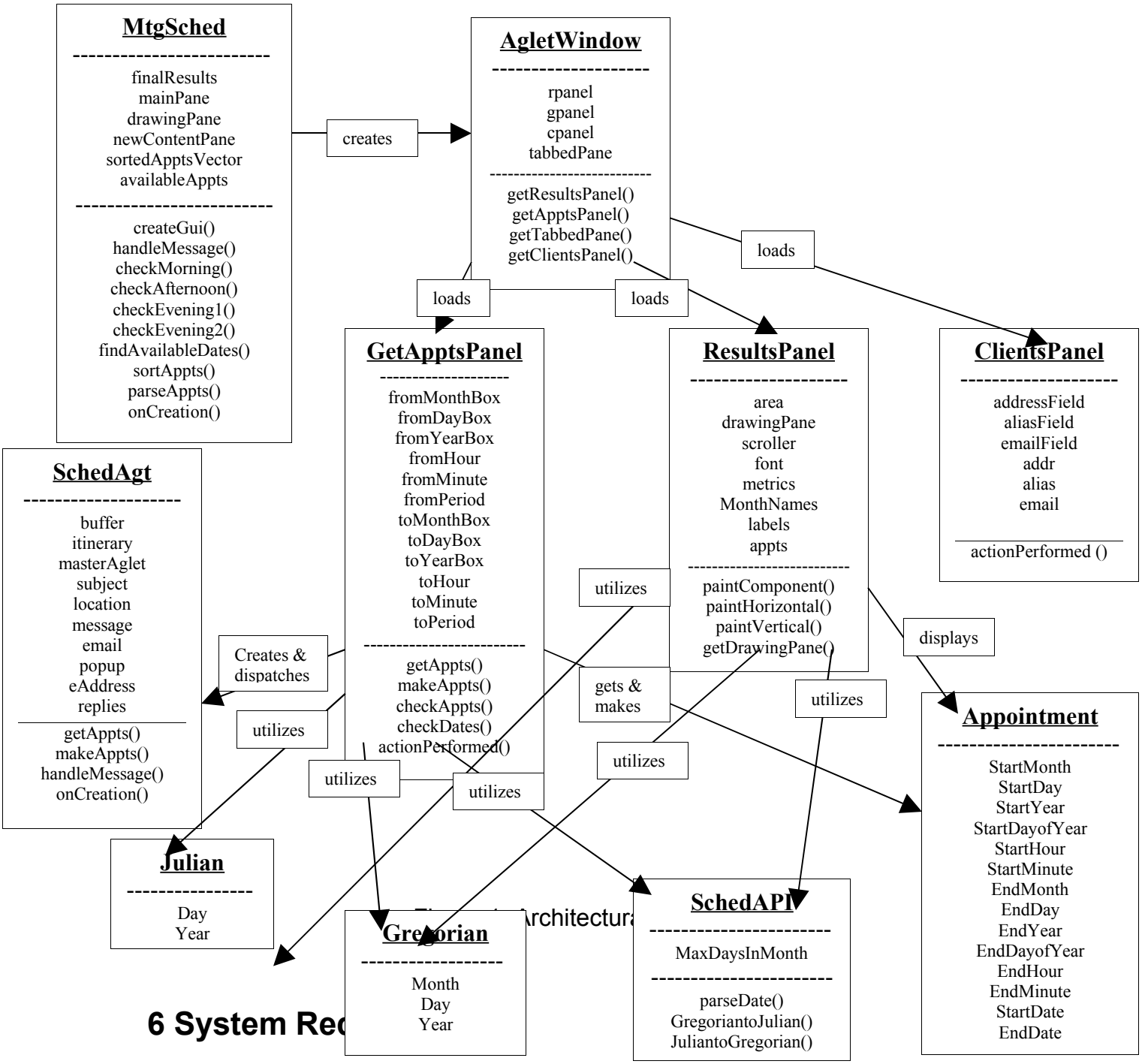
4.4 Testing

Testing is the most important part of developing a product. The author performed testing on three personal computers connected on a local area network. Each machine played the role of employee machine and manager machine. This meant that each machine would have the chance to run the product. The first computer was a 700Mhz Intel Celeron machine with 512 megabytes of RAM running Microsoft Windows XP Professional and Outlook 2000. The second machine was a 2.0Ghz Intel Celeron laptop with 256 megabytes of RAM running Microsoft Windows XP Home Edition and Outlook 2000. The third machine was a 2.66Ghz Pentium 4 with 128MB of RAM running Microsoft Windows XP Home Edition and Outlook 2000.

Other users were allowed to use the software and were encouraged to try "off the wall" things that could break the software. I also allowed the users to give me feedback and give me advice on how the software could be improved.

5 High Level Architectiral Diagram

Figure 1 contains a high level diagram of the Meeting Scheduler architecture.



6 System Requirements

This application was written using Java SDK Version 1.4.1. It will be developed on a 700 Mhz Intel Celeron processor system running Windows XP Professional. Now that the product is finished, the system requirements were able to be more accurately calculated. The system requirements are as follows.

System:	133 MHz or higher Pentium-compatible CPU
RAM:	64 MB or more

Operating System:	Windows 98, Windows Me, Windows 2000, Windows XP Home Edition, Windows XP Professional
Hard Drive Space:	At least 15 Mb of free space

7 Meeting Scheduler Software

The Meeting Scheduler software has one main window containing three tabs. The first is the Get Appointments tab. This is where the majority of the work is accomplished when using the software. On this tab the user can retrieve and make appiontments as well as change details (start/end date, subject, location). The second tab is the “View Results” tab. Here the user can view a graphical representation of all of the clients’ schedules. Nothing is displayed on this screen until the user retrieves appointments. The final tab is the “Setup” tab. This tab is used to add and remove clients whose schedules will be used in the scheduling process.

7.1 Get Appointments Tab

Figure 2 shows a screen shot of the Get Appointments Tab.

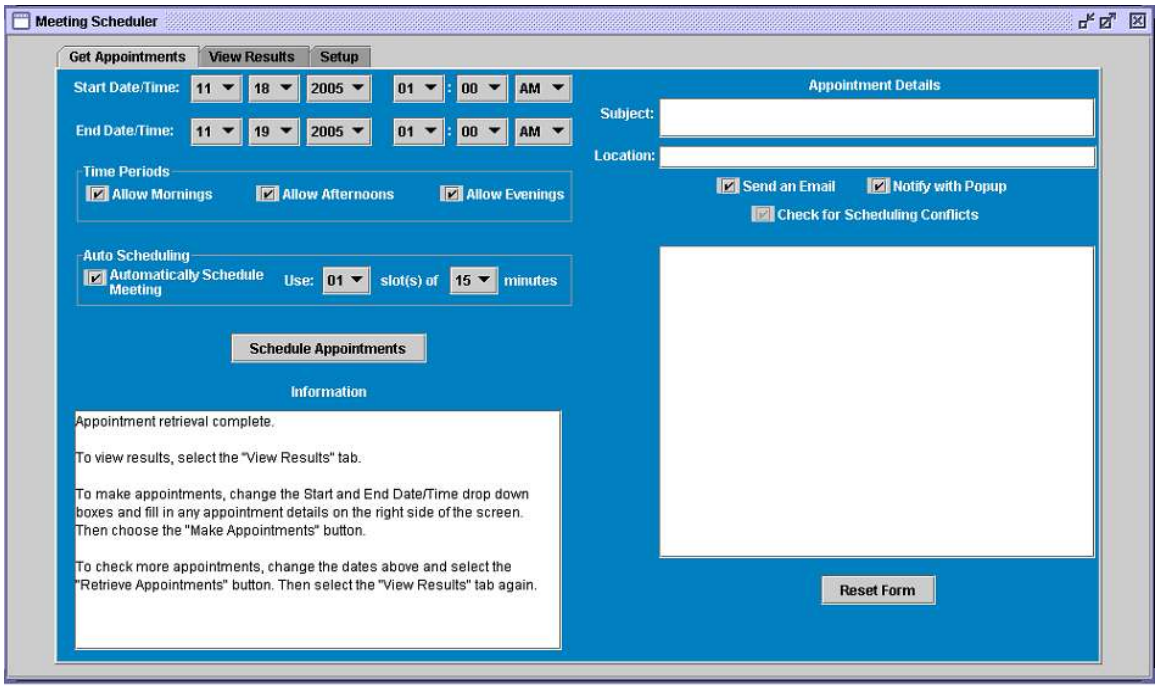


Figure 2: Get Appointments Tab

7.2 View Results Tab

Figure 3 shows a screen shot of the View Results Tab.

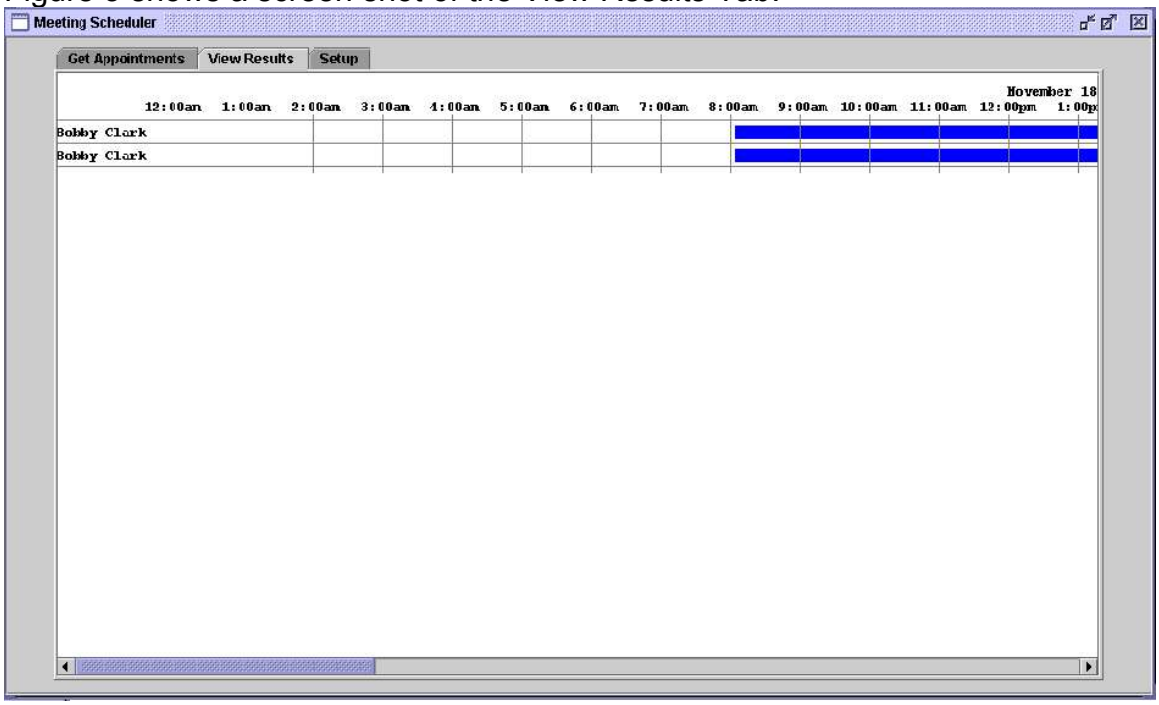


Figure 3: View Results Tab

7.3 Setup Tab

Figure 4 shows a screen shot of the Setup Tab.

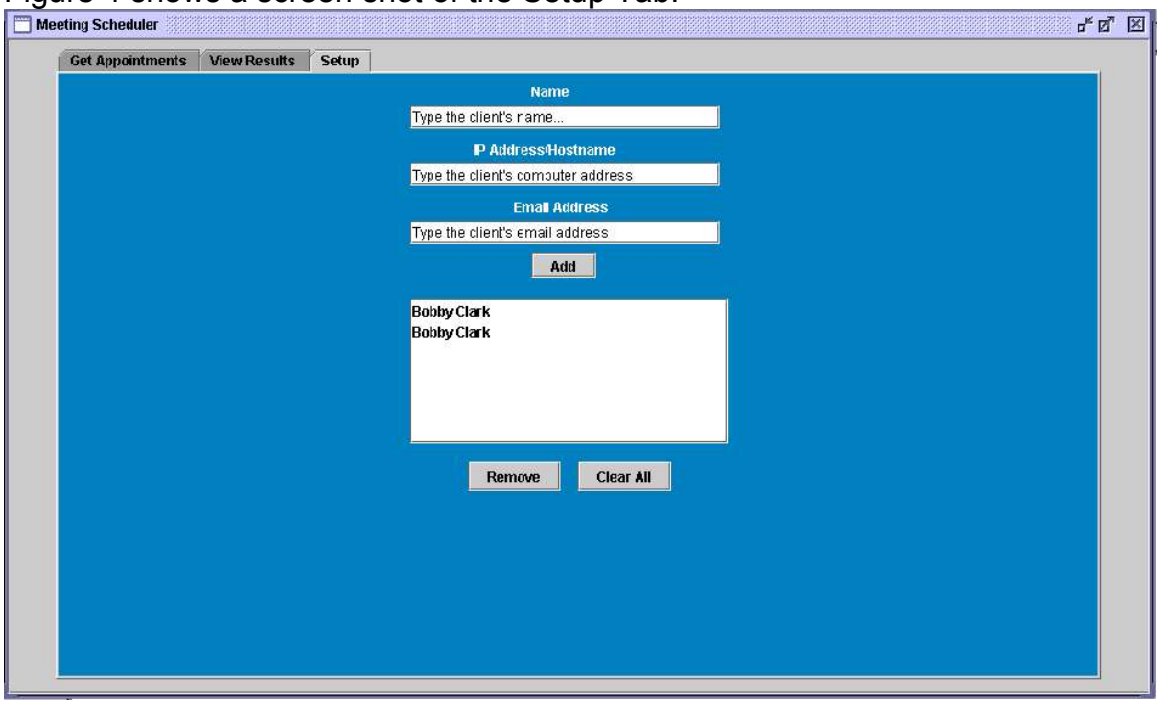


Figure 4: Setup Tab

8 Future Enhancements and Improvements

The future is wide open for this project. There are certain things that I think would greatly improve this software. More extensive help would be the first thing to enhance. Another thing that would be good for below average users is some kind of wizard that walks the user through the scheduling process using a series of dialogs. It would be similar to an Installshield installation. Making this meeting scheduler work with a web interface as opposed to a local application has many benefits. Those benefits include a smaller installation and fewer pieces to update when maintenance needs to be applied.

Some smaller scale improvements could also be implemented while leaving most of the current architecture in place. Things like allowing the user to select a current client and view its IP address and email address. Also, if the scheduling aglet visits a client and the client does not respond to the popup question, the scheduling aglet may wait indefinitely. It would be nice to put a timer on that action. The email responsibility can also be shifted from the clients' Outlook to the Meeting Scheduler itself. The Meeting Scheduler could also be changed to allow the user to pick from a set of appointment times that where found to be available by the software.

References

Cockayne, William T. and Michael Zyda. *Mobile Agents*. Greenwich, CT: Manning Publications Co, 1998.

Jing, Jin and Abdelsalam Helal and Ahmed Elmagarmid. "Client-Server Computing in Mobile Environments." *ACM Computing Surveys* **(June 1999): 117-157.**

Kun, Yang and Guo Xin and Liu Dayou. "Security in Mobile Agent System: Problems and Approaches."

Lange, Danny B. and Mitsuru Oshima. "Seven Good Reasons for Mobile Agents." *Communications of the ACM* **(March 1999): 88-89.**

Sundsted, Todd. "An introduction to agents." ***JavaWorld*****. June 1998. http://www.javaworld.com/javaworld/jw-06-1998/jw-06-howto.html (07 Mar. 2004).**

Wang, David K. and James K. Wang. "Towards the Distributed Processing of Mobile Software Agents."

Wong, David and Noemi Paciorek and Dana Moore. "Java-Based Mobile Agents." *Communications of the ACM* **(March 1999): 92-102.**