P-1559

MOBILE AGENTS IN WIRELESS DEVICES

A PROJECT REPORT

Submitted by

AARTHI.J 71201205001

DHIVIYA.D 71201205011

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2005

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**MOBILE AGENTS IN WIRELESS DEVICES**" is the bonafide work of "**Ms.AARTHI.J - 71201205001** and **Ms.DHIVIYA.D - 71201205011**" who carried out the project work under my supervision.

[illegible]	[illegible]
SIGNATURE	**SIGNATURE**
Dr.S.Thangasamy	**Mrs.N.Chitra Devi**
HEAD OF THE DEPARMENT	**SUPERVISOR**
	Senior Lecturer,
Dept of Computer Science and Engg.	Dept of Information Technology
Kumaraguru College of Technology	Kumaraguru College of Technology
Coimbatore – 641 006	Coimbatore – 641 006
[illegible]	[illegible]
INTERNAL EXAMINER	EXTERNAL EXAMINER
	18/04/2005.

ACKNOWLEDGEMENT

We express our sincere thanks to **Dr.K.Arumugam, B.E (Hons), M.S. (USA), MIE.,** Correspondent, Kumaraguru College of Technology and the management, for proving us an opportunity to undertake this project work.

We express our profound gratitude to **Dr.K.K.Padmanabhan B.Sc. (Engg)., M.Tech., Ph.D.,** Principal, Kumaraguru College of Technology, Coimbatore, for permitting us to undergo a project work.

We are greatly indebted to **Dr. S. Thangasamy, Ph.D.**, Professor and Head of the Department of Computer Science and Engineering, for the immense support he has provided us throughout our project.

We extend our sincere thanks to our project coordinator **Prof.K.R.Baskaran, B.E., M.S.,** Assistant Professor, Department of Information Technology, for his constant support and encouragement.

We would like to express our heart felt thanks to our guide and Class advisor, **Mrs.N.Chitra Devi, M.E.,** Senior Lecturer, Department of Information Technology, for her everlasting counseling and untiring help throughout our project.

We like to express our deep sense of gratitude to our families and friends whose help aided the successful completion of the project.

DECLARATION

We,

AARTHI.J	71201205001
DHIVIYA.D	71201205011

Declare that the project entitled "**MOBILE AGENTS IN WIRELESS DEVICES**", submitted in partial fulfillment to Anna University as the project work of Bachelor Of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and guidance of **Mrs.N.Chitra Devi, M.E.**, Senior Lecturer, Department of Information Technology , Kumaraguru College Of Technology, Coimbatore.

Place: Coimbatore

Date : 18 · 04 · 2005

[illegible]

[Aarthi.J]

[illegible]

[Dhiviya.D]

Project Guided by

[illegible]

[Mrs.N.ChitraDevi, M.E.,]

ABSTRACT

This project **'MOBILE AGENTS IN WIRELESS DEVICES'** is a prototype agent – based framework that minimizes the load on wireless link and supports disconnected operations in connections between a mobile device and a remote host.

The user accesses Remote hosts through a GPRS enabled handheld device. The device initially submits the request to the Server, which in turn processes it using the Mobile Agent platform. The Agent transports itself to the Remote host to fetch the results back to the Server. Once the results are available at the Server, the user can re-connect to download the results. This is implemented in a Book Search Application.

The objective of this project is that it reduces communications over wireless links to overcome low bandwidth and network disconnection. In addition, it enhances service functionality by operating without constant user input. Finally, the system is platform independent.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

• SRS	: Software Requirement Specification
• SDK	: Java Development Kit
• API	: Application Programming Interface
• J2ME	: Java 2 Micro Edition
• GUI	: Graphical User Interface
• MIDP	: Mobile Information Device Protocol
• WMA	: Wireless Messaging API
• MMAPI	: Mobile Media API
• SMS	: Short Messaging Service
• ASDK	: Aglet Software Development Kit
• ATP	: Agent Transfer Protocol
• ATCI	: Agent Transfer and Communication Interface
• ISBN	: International Standard Book Number
• HTTP	: Hyper Text Transfer Protocol
• GPRS	: General Packet Radio Service
• IDE	: Integrated Development Environment
• URL	: Universal Resource Locator

1. INTRODUCTION

1.1 OBJECTIVE:

The objective of this project is to employ Mobile Agents to provide a prototype agent based framework that minimizes the load over wireless links and supports disconnected operations. This is achieved by implementing a Book Search Agent application.

1.2 PROBLEM DESCRIPTION:

The Project aims at providing an optimal solution for the potential problems involved in Network Data Transfer.

The main problem involved in communication from a Mobile Device to Remote Host is the time that that the User has to remain connected to the network. In conventional systems, the User remains connected to the GPRS network through out the Search process.

The second problem is the Network congestion encountered during the Data transfer operation. Conventional systems suffer from low bandwidth and high load in the process of operation.

The next problem is the Support for Disconnected operations. Since the User remains connected to the network for a longer duration there is every possibility for network disconnection. The result availability cannot be guaranteed on re-connection.

Finally the system to be developed should be Platform independent to cooperate with geographically separated systems.

2. SYSTEM AND SOFTWARE OVERVIEW

2.1 EXISTING SYSTEM AND ITS LIMITATIONS:

The existing system connects the Mobile device through a GPRS connection to the Remote host. The Middle layer in this 3-tier architecture is the Server, which directly looks up in the database of the remote host. The connection is alive throughout the time of data transmission from the mobile device to the Remote host and back.

LIMITATIONS:

- Requires the Mobile device to remain connected throughout the process increasing the total cost of operation
- Inability to operate in Low Bandwidth
- High incidence of network errors

2.2 PROPOSED SYSTEM AND ITS ADVANTAGES:

The proposed system introduces an Agent Platform between the Server and the Remote Host. The GPRS connection is closed once the User submits the request to the Server. The Server in turn submits the request to the listening Agent, which retrieves requested information from the Remote host. The result availability is announced to the User through an SMS, who then reconnects to download the results.

ADVANTAGES:

- Reduces communication over wireless links to overcome bandwidth and network disconnection
- Enhances service functionality by operating without constant user input
- Platform independant

2.3 JAVA 2 MICRO EDITION

The J2ME Wireless Toolkit supports the development of Java applications that run on devices compliant with the Mobile Information Device Profile (MIDP) version 2.0, such as cellular phones, two-way pagers, and communicators. In addition, the Wireless Toolkit supports development of applications compliant with the Wireless Messaging API (WMA) and the Mobile Media API (MMAPI).

Features of the Wireless Toolkit:

The KToolBar, included with the J2ME Wireless Toolkit, is a minimal development environment with a graphical user interface (GUI) for compiling, packaging, and executing MIDP applications. There is a need for is a third-party editor for Java source files and a debugger.

An IDE compatible with the J2ME Wireless Toolkit, such as the Sun TM One Studio 4, Mobile Editor, provides even more convenience. While using the Wireless Toolkit within an IDE, it is possible to edit, compile, package, and execute or debug MIDP applications, all within the same environment.

Compiling, Preverifying, and Debugging:

When compiling MIDlets through the KToolBar (or an IDE compatible with the toolkit), the source files are compiled using the J2SETM SDK compiler. Preverification of the compiled files is done with the Preverifier that prepares class and JAR files and class directories. Preverification takes place immediately after compilation. It is possible to debug applications within the environment using the Emulator, which simulates the execution of the application on various devices.

Packaging:

It is possible to package your MIDlet suite from the KToolBar or with a compatible IDE.The KToolBar gives the choice of creating a standard package or creating an obfuscated package that produces a smaller JAR file by reducing the size of the classes in the suite through the obfuscation process.

Authenticating and Authorizing MIDlets:

Trusted applications can be created that have permission to use protected APIs. Permission can also be requested to access network protocol APIs through the Project Settings dialog box from the KToolBar. The MIDlet suite can be signed and assigned a security domain that defines the suite's authorization level with the Sign MIDlet Suite window.

Performance Tuning:

The Wireless Toolkit's Profiler enables us to optimize the performance of the MIDlet suite by determining where bottlenecks might be occurring during runtime execution time of the MIDlet suite can be improved by examining the time spent in method calls, the number of times a method is called during runtime, and the amount of time a method runs compared to the overall runtime of the application. The performance speed of the application can be adjusted in the Performance panel of the Project Settings dialog box. Setting the speed features does not demonstrate how the application would run on an actual device; however, by adjusting the speed emulation parameters, a better representational performance of the application on a device can be achieved.

Memory and Network Monitoring:

The Wireless Toolkit provides tools to examine and analyze memory usage by the application and network transmissions between the device and the network. It is possible to get an overall view of memory usage during runtime of the application and get a breakdown of memory usage per object to see where in the application memory usage can be optimized.

With the Network Monitor, network transmissions for several types of network protocols can be examined.

Working With the Emulator:

The J2ME Wireless Toolkit comes with a selection of emulated devices to run and test the applications on. Representations of mobile devices are available from the Device list on the KToolBar. The emulated devices are capable of emulating the features in the CLDC, MIDP, MMAPI, and WMA specifications. The functionality for an emulated device through the Preferences window can be set. Various emulator utilities such as the Profiler, the Network Monitor, the Memory Monitor, and the Certificate Manager from the Utilities window are also present.

2.4 MOBILE AGENTS

A mobile agent is a program that can migrate from host to host in a network of heterogeneous computer systems and fulfill a task specified by its owner. It works autonomously and communicates with other agents and host systems. During the self-initiated migration, the agent carries all its code and the complete execution state with it. Mobile agent systems build the environment in which mobile agents can exist.

Migration of agents is based on an infrastructure that has to provide the necessary services in the network. The infrastructure is a set of agent servers that run on platforms within a possibly heterogeneous network. Each agent server hides the vendor specific aspects of its host platform and offers standardized services to an agent that is docking on to such a server concluding migration. Services include access to local resources and applications, e.g. traditional web-servers, the local exchange of information between agents via message passing, basic security services, creation of new agents, etc.

Mobile agents are defined in formal terms by computer scientists, as, objects that have ***behavior*, *state*,** and ***location***. A subset of behaviors of every agent is inherited from the model, notably those behaviors that define the means by which agents move from place to place. Mobile agent models almost always define a method of interagent messaging as well. Finally, a mobile agent model is not complete without defining a set of events that are of interest to the agent during its lifetime. The set of events varies a bit from model to model, but the following is a list of the most common ones:

- **Creation** -- Analogous to the constructor of an object. A handler for this event should initialize state and prepare the agent for further instructions.
- **Disposal** -- Analogous to the destructor of an object. A handler for this event should free whatever resources the agent is using and prepare the agent for burial.
- **Dispatch** -- Signals the agent to prepare for departure to a new location. The agent itself upon requesting to migrate can generate this event explicitly, or another agent that has asked this agent to move can trigger it.

- **Arrival** -- Signals the agent that it has successfully arrived at its new location and that it should commence performing its duties.
- **Communication** -- Notifies the agent to handle messages incoming from other agents and is the primary means of inter-agent correspondence.

2.5 AGLETS WORKBENCH

The Aglets Workbench, developed at IBM's research labs in Japan, is aimed at producing stand-alone mobile agents. The complete package offers a graphical environment for building mobile agent applications in Java, an agent server, and the specification for an Agent Transfer Protocol (ATP). The Aglets Workbench supports both mobility and itinerary.

2.5.1. AGLETS

The **Aglet** represents the next leap forward in the evolution of executable content on the Internet, introducing program code that can be transported along with state information. Aglets are Java objects that can move from one host on the Internet to another. That is, an aglet that executes on one host can suddenly halt execution, **dispatch** itself to a remote host, and resume execution there. When the aglet moves, it takes along its program code as well as its data. A built-in security mechanism makes it safe to host untrusted aglets.

The major System Goals of Aglets are

- Provide an easy and comprehensive model for programming mobile agents without requiring modifications to Java VM or native code.

- Support dynamic and powerful communication that enables agents to communicate with unknown agents as well as well-known agents.
- Design a reusable and extensible architecture.
- Design a harmonious architecture with existing Web/Java technology.

2.5.1.1 System Architecture

The Aglets architecture consists of two APIs and two implementation layers.

- Aglet API
- Aglets Runtime Layer - The implementation of Aglet API
- Agent Transport and Communication Interface
- Transport Layer

The Aglets runtime layer is the implementation of Aglet API, which provides the fundamental functionality such as creation, management or transfer of aglets. This layer defines the behavior of APIs such as Aglet and AgletContext, and can serve multiple AgletContext objects.

The transport layer is responsible for transporting an agent to the destination in the form of a byte stream that contains class definitions as well as the serialized state of the agent. This layer is also defined as an API, called Agent Transfer and Communication Interface (ATCI), which allows the Aglets runtime to use the transport layer in a protocol-independent manner. The implementation of ATCI is responsible for sending and receiving an agent and establishing a communication between agents. The current Aglets implementation uses the Agent

Transfer Protocol (ATP), which is an application-level protocol for transmission of mobile agents. ATP is modeled on the HTTP protocol, and can be used to transfer the content of an agent in an agent-system-independent manner. To enable communication between agents, ATP also supports message-passing.

When an aglet issues a request to dispatch itself to a destination, the request travels down to the Aglets runtime layer, which converts the aglet into the form of a byte array consisting of its state data and its code. If the request is successful, the aglet is terminated, and the byte array is passed to the ATP layer through the ATCI. The ATP, which is the implementation of ATCI, then constructs a bit stream that contains general information such as the agent system name and agent identifier, as well as the byte array from the Aglets runtime.

2.5.1.2 Aglet Object and Its Life Cycle

The Aglets class provides the basic functionality for a mobile object, and every mobile object (aglet objects) has to be an instance of a subclass of the com.ibm.aglet.Aglet class. To be useful, an aglet has to be instantiated first. There are two ways to create a new instance of an aglet. The first is to instantiate a completely new aglet from class definitions by calling AgletContext.createAglet(URL codebase, String name, Object init). This primitive creates a new instance within the specified context and initializes it if necessary, then invokes Aglet.onCreation(Object init) on the created object along with the initializer object passed to the createAglet primitive. The other way is to create a copy of an existing aglet by using the Aglet.clone() primitive. The cloned aglet has the same state as the original one but has a different AgletID object, an thus a distinct identity.

Once created, an aglet object can be dispatched to and/or retracted from a remote host, deactivated and placed in secondary storage, then activated later.

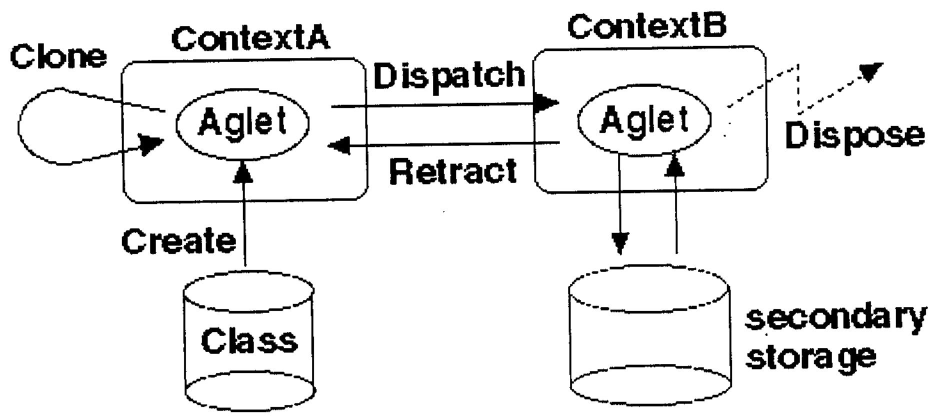


Figure 1. Aglet Life Cycle

An aglet can dispatch itself to a remote host by calling the Aglet.dispatch(URL dest) primitive. To be more precise, an aglet occupies the aglet context and can move from this context to others during its execution. Because the runtime system may serve multiple contexts within one Java VM, these contexts can be in the same VM. Dispatching causes an aglet to suspend its execution, serialize its internal state and bytecode into the standard form and then to be transported to the destination. On the receiver side, the Java object is reconstructed according to the data received from the origin, and a new thread is assigned and executed.

Aglets support persistency of aglet objects. All mobile aglet objects can be persistent in nature, because they must be convertable into a bit-stream; consequently, this stream can be stored in secondary storage.

Unlike normal Java objects, which are automatically released by garbage collector, an aglet object, since it is active, can decide whether or not to die. If you call the dispose() method to kill the aglet, onDisposing() is called to perform the finalization suitable for the current state of the aglet.

2.5.1.3 Agent Transfer Protocol (ATP)

The ATP protocol is based on a request/response paradigm between agent services. Agent service *A* establishes a connection with agent service ***B***, then sends a request to ***B*** and waits for the response. Thus, *A* acts as a ***sender*** (of the request) and ***B*** acts as a ***recipient***.

ATP defines four standard request methods for agent services:

- Dispatch
- Retract
- Fetch.
- Message

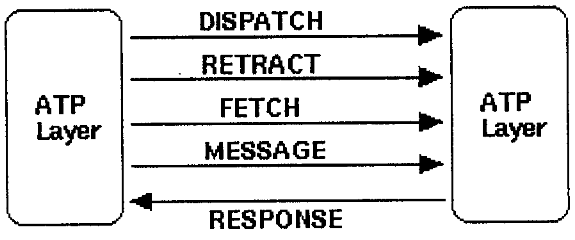


Figure 2. Communication through ATP

2.5.2 Tahiti Server

Tahiti is an application program that runs as an agent server. Multiple servers (**Tahiti**) can be run on a single computer by assigning them different port numbers. **Tahiti** provides a user interface for monitoring, creating, dispatching, and disposing of agents and for setting the agent access privileges for the agent server.

All the agents are displayed in the list box in **Tahiti**. By selecting one of the list items, the corresponding agent can be controlled.

- **Create:** Creates a new aglet. A dialog window for the aglet's URL specification will appear.
- **Dialog:** Sends a request to an aglet to open its dialog panel. (When this button is clicked, an onDialog () message is sent to the aglet)
- **AgletInfo:** Shows the properties of the agent.
- **Dispose:** Destroys the agent.
- **Clone:** Make a copy of the agent. Cloned agent runs on the same context.
- **Dispatch:** Sends the agent to another server. After dispatching agent, original agent on the current server does no more exist on the current server. The protocol for the destination URL is Agent Transfer Protocol. It is specified as:

  atp://aglets.trl.ibm.com:434

- **Retract:** Retracts (draw backs) a dispatched agent from a remote server. First the target server is specified and a list of agents on the target server will be provided. Then, one of the aglets from the server can be selected.

2.6 APACHE TOMCAT 5.0

Tomcat 5 implements the Servlet 2.4 and JavaServer Pages 2.0 specifications from the Java Community Process, and includes many additional features that make it a useful platform for developing and deploying web applications and web services.

2.6.1 Servlets

A Servlet is a Java class used to extend the capabilities of servers that host applications accessed via a request-response programming model. Servlets are not tied to a specific client-server protocol but they are most commonly used with HTTP.

A Servlet Engine provides the runtime environment in which a servlet executes. It manages the life cycle of servlets from when they are first created through to their imminent destruction. It executes within a Java Virtual Machine.

The general rules to be followed while writing a HTTP Servlet:

- Extend HttpServlet class, which implements the Servlet interface
- Override doGet () - to handle GET requests. Override doPost () method to handle POST requests.
- Within the doGet/Post method, an HttpServletRequest object represents user's request and an HttpServletResponse object represents response to the user.
- The response can be written to the user through PrintWriter object obtained from the HttpServletResponse object.

3. REQUIREMENTS ANALYSIS

3.1 SOFTWARE REQUIREMENTS SPECIFICATION

3.1.1. Introduction

3.1.1.1 Purpose:

The purpose of this document is to specify the requirements of project "MOBILE AGENTS IN WIRELESS DEVICES", an agent – based framework that minimizes the load on wireless link and supports disconnected operations in connections between a mobile device and a remote host.

3.1.1.2 Scope:

SRS forms the basics for agreement between the client and the supplier and what the software product will do. It also provides a reference for the validation of the final project.

Any changes made to the SRS in the future will have to go through formal change approval process.

3.1.1.3 Definition:

Customer:

A person or organization, internal or external to the producing organization, who takes financial responsibility for the system. In a large system this may not be the end user. The customer is the ultimate recipient of the developed product and its artifacts.

User:

A person who uses the Developed System.

Analyst:

The Analyst details the specification of the system's functionality by describing the requirements aspect and other supporting software requirements

3.1.1.4 Abbreviation:

SRS	:	Software Requirement Specification
JDK	:	Java Development Kit
J2ME	:	Java 2 Micro Edition
MIDP	:	Mobile Information Device Protocol
SMS	:	Short Messaging Service
ASDK	:	Aglet Software Development Kit
ATP	:	Agent Transfer Protocol

3.1.2. General Description

2.1Product Overview:

The project aims to connect the Mobile User to the Remote host Database through the Aglet Platform. The Agent migrates to the Remote host through a process called Dispatching. It then performs the r requested search operation local to the Remote Database. This helps reduce the active connection time of the user being connected.

3.2.2.2 User Characteristics:

The Users of this system include the Common public who wish to access the Data from a geographically distant host.

3.2.2.3 General Constraints:

The project platform is independent of any operating system since the entire code is written in Java, a platform independent language.

3.2.2.4 General assumptions:

The User has a GPRS enabled mobile phone and has good GPRS coverage in order to use this system.

3.3.3. Specific Requirements

3.3.3.1 Inputs and Outputs:

The User Interface of the System is provided by the MIDlet. The MIDlet prompts for Search and download options. Depending on the options selected, the data corresponding to the Search key is entered by the User. The final output is the display of Book Information from the Database.

3.3.3.2 Functional requirements:

- The system should be able to initiate the Search operation with the Data.
- The system should display the desired output in the mobile screen and the data should not get lost during transmission.

3.3.3.3 System Requirements:

Hardware Requirements:

Processor	:	Pentium IV
RAM size	:	128 MB RAM
Hard disk capacity	:	20GB

Software Requirements:

Operating System	:	Windows 2000/XP
Database Used	:	Microsoft Access
Language	:	J2SDK 1.4.2_04

Mobile Simulator	: J2ME Wireless Toolkit
Server	: Apache Tomcat 5.0
Agent Platform	: Aglet SDK

3.3.3.4 Performance constraints:

The system should be Capable of supporting disconnected operations in the Internet. The connection time is greatly reduced on deploying Agents to carry out the Search operation.

3.3.3.5 Software Constraints:

The User mobile needs to be Java enable and capable of supporting MIDP2.0 protocol.

The Communication Manager requires that J2SDK 1.4.0.2 with Apache Tomcat Server be installed and running in the server system. The Aglet SDK is to be installed on the Agent Gateway.

The Remote host also has to support the Aglet platform.

4. SYSTEM DESIGN

4.1 DESIGN DOCUMENTS

4.1.1 Use – Case Diagram

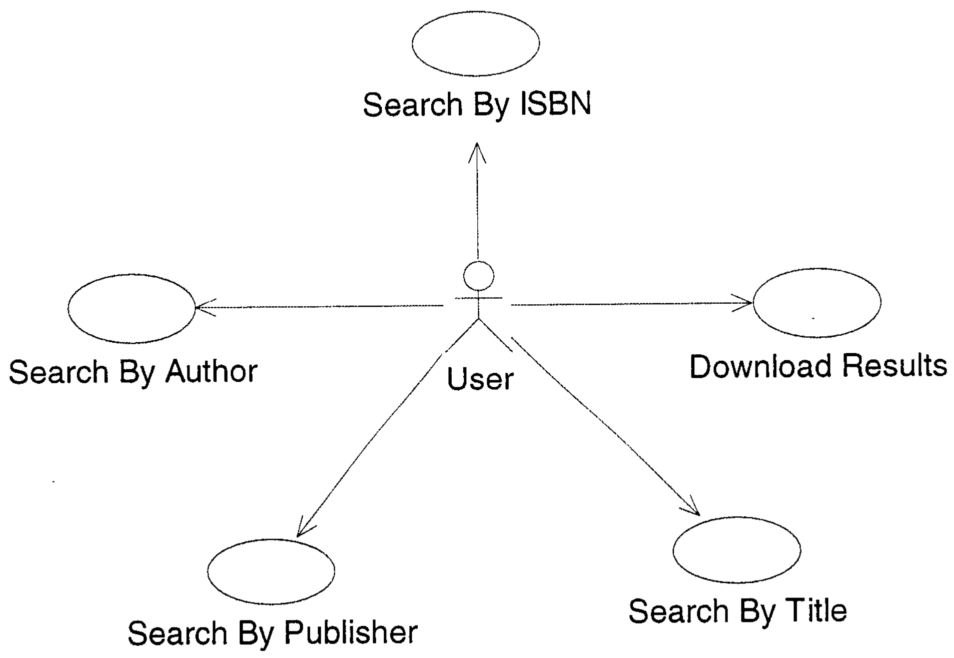


Figure 3. Use Case Diagram

4.1.2 Sequence Diagram

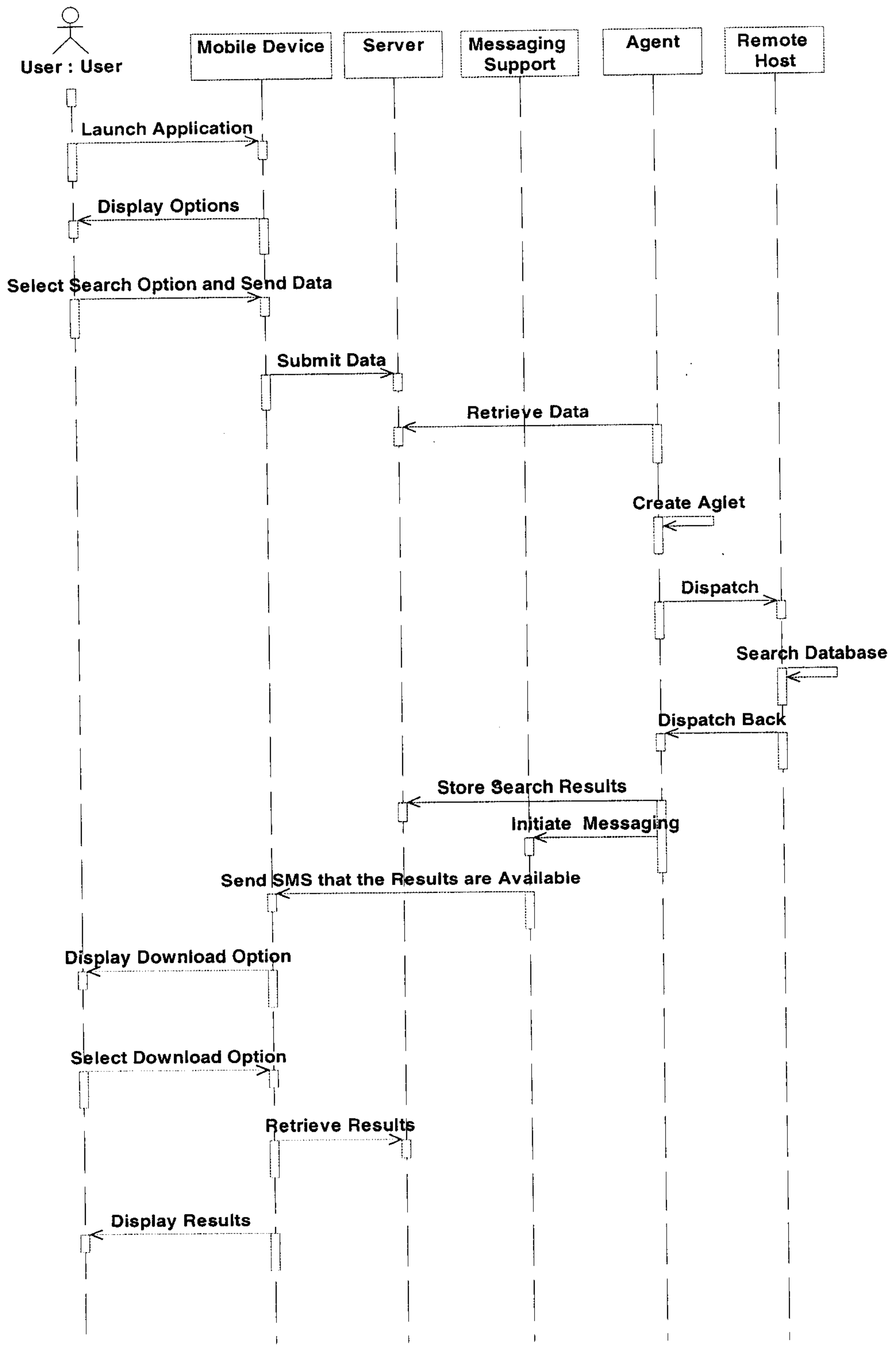


Figure 4. Sequence Diagram

This diagram explains the Sequence flow of the Search Application

4.1.3 Collaboration Diagram

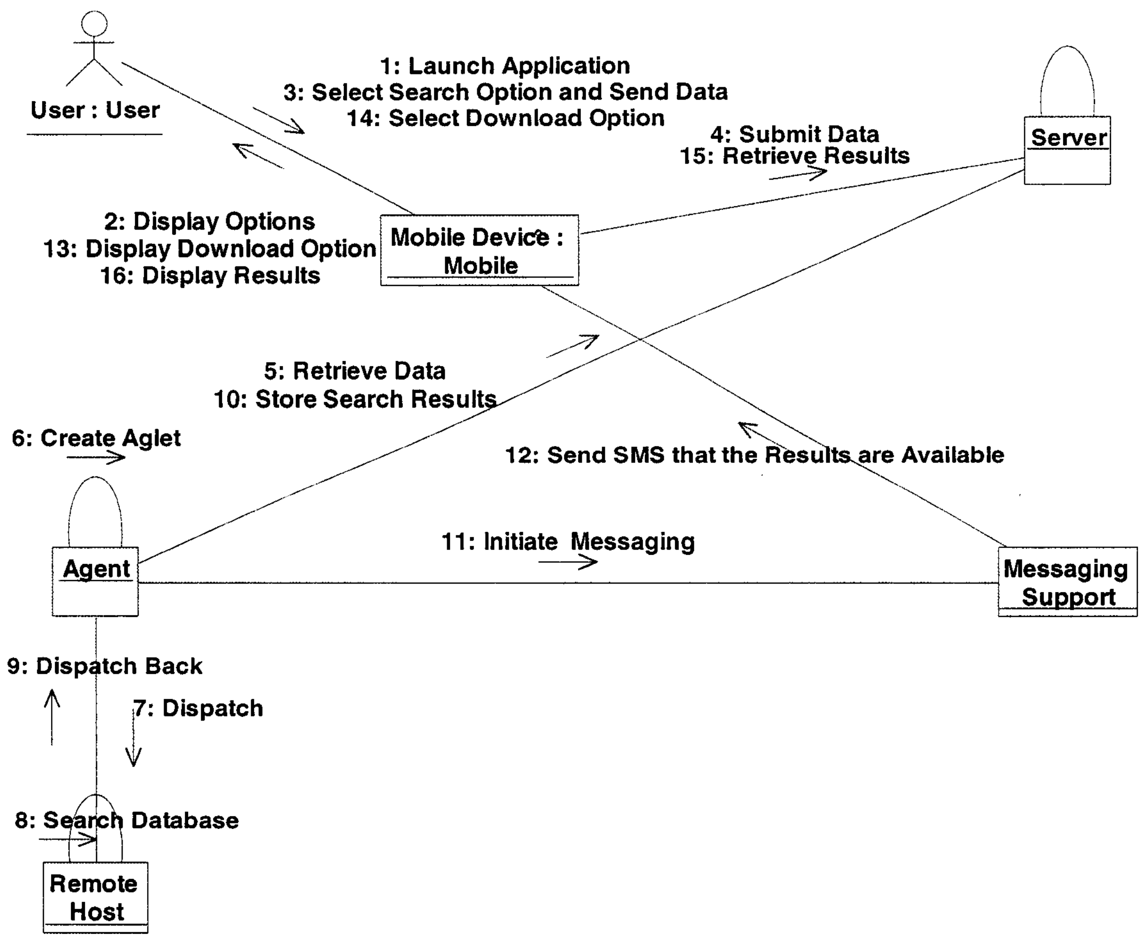


Figure 4. Collaboration Diagram

This diagram explains the Collaboration between different components of the Search Application

4.1.4 Class Diagram

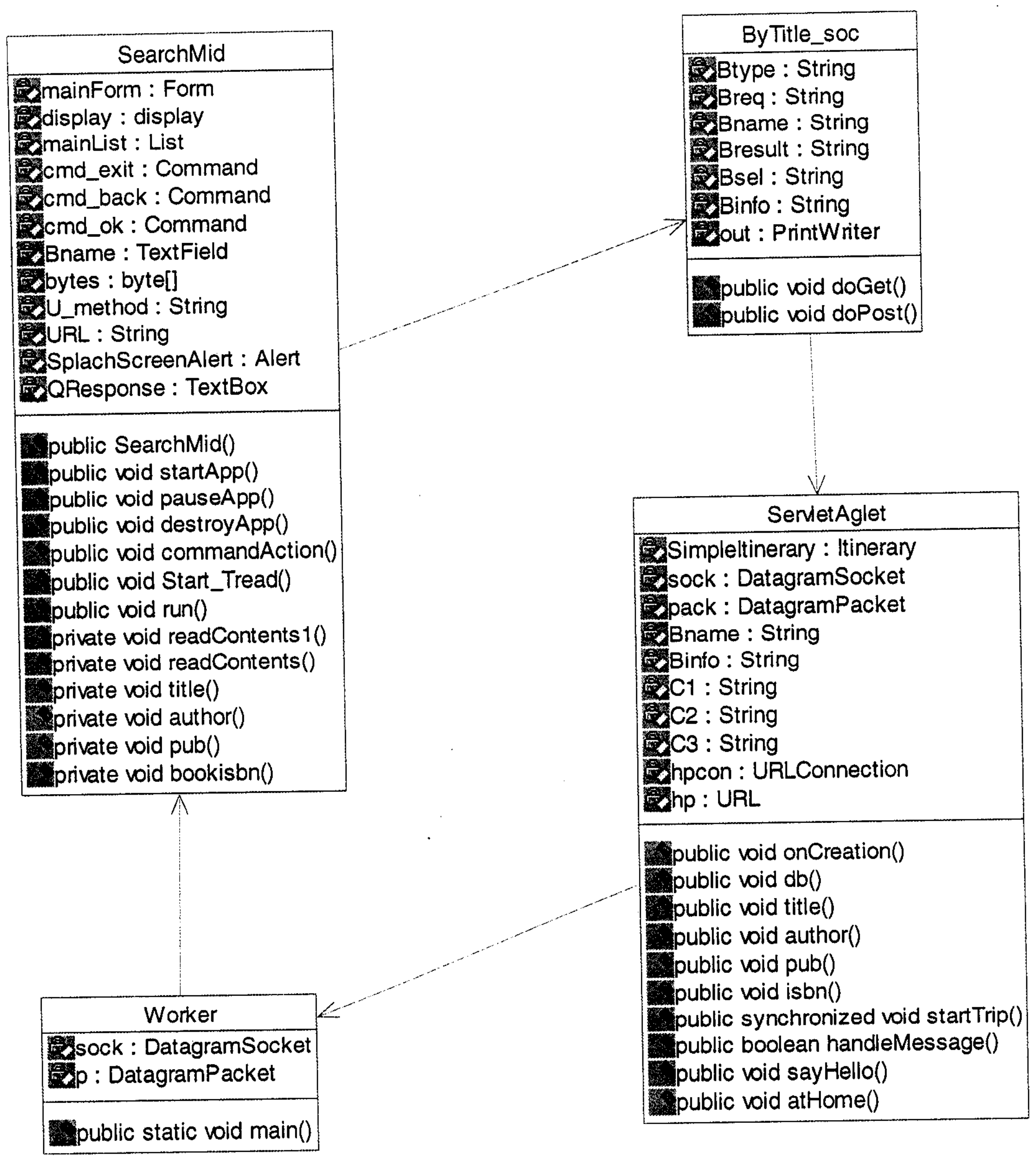


Figure 4. Class Diagram

4.1.5 Database Design

4.1.5.1 Table Design

TABLE NAME: PUBLISHERS

S.NO	FIELD NAME	DATATYPE
1.	PubID	Auto Number
2.	Name	Text
3.	Company Name	Text
4.	Address	Text
5.	City	Text
6.	State	Text
7.	Zip	Text
8.	Telephone	Text
9.	Fax	Text
10.	Comments	Memo

TABLE NAME: TITLES

S.NO	FIELD NAME	DATATYPE
1.	Title	Auto Number
2.	Year Published	Number
3.	ISBN	Text
4.	PubID	Number
5.	Description	Text
6.	Notes	Text
7.	Subject	Text
8.	Comments	Memo

TABLE NAME: TITLE AUTHOR

S.NO	FIELD NAME	DATATYPE
1.	ISBN	Text
2.	Au_ID	Number

TABLE NAME: AUTHORS

S.NO	FIELD NAME	DATATYPE
1.	Au_ID	Auto Number
2.	Author	Text
3.	Year Born	Number

4.1.5.1 Table Relationship

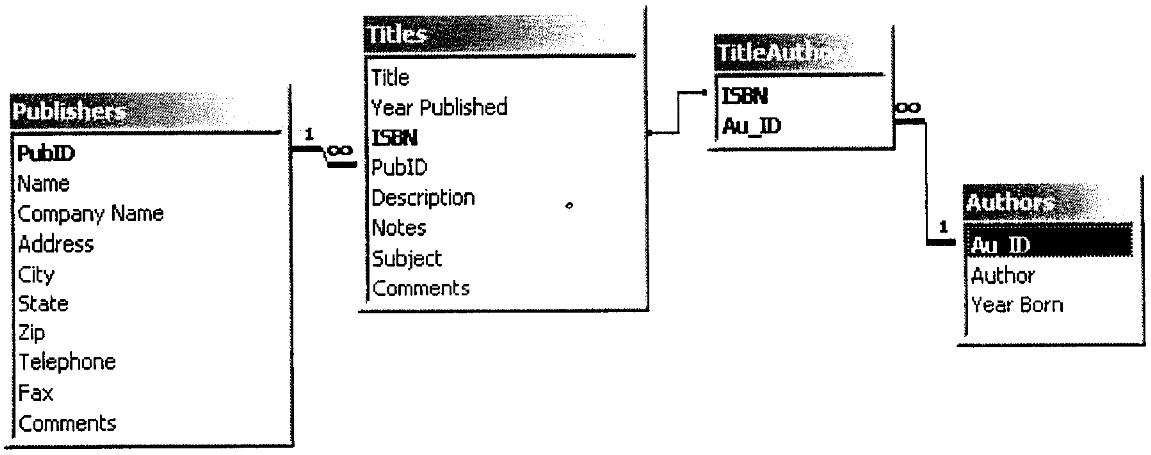


Figure 6. Table Relationships

5. SYSTEM IMPEMENTATION

5.1 OPERATING PRINCIPLE

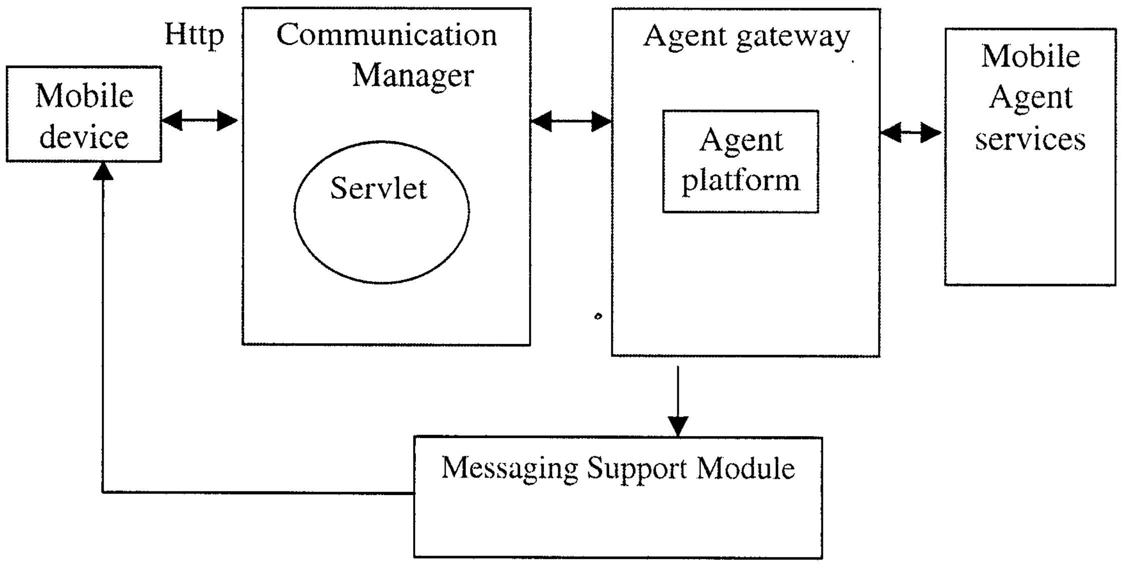


Figure 7. Operating Principle

5.2 MODULE DESCRIPTION

5.2.1 User Interface

The User Interface is programmed using the J2ME Wireless Toolkit 2.0.The motive behind using this software, is the Emulator, which is a Mobile Interface Simulator. The Emulator provides user-interface functionalities typical to a Mobile Device.

The User Interface programming begins with the coding of MIDlets. A MIDlet is a MID Profile application. The application must extend this class to allow the application management software to control the MIDlet and to be able to retrieve properties from the application descriptor and notify and request state changes. The methods of this class allow the application management software to create, start, pause, and destroy a MIDlet. A MIDlet is a set of classes designed to be run and controlled by the application management software via this interface.

The MIDlet employed in the project employs the basic functionalities as the Top-level option structure for a Book Search Application,

- Initiating a New Search
- Downloading the Results

When the New Search is initiated, the MIDlet takes in charge of taking in the Search Criteria. There are four options deciding the key using which the search is to be performed. They are,

- Title
- Author
- Publisher

- ISBN

The User selects one of the above options and enters the required information in the Text Area that follows. This is submitted to the Server through HTTP Connection. After submission, the Mobile is disconnected from the network.

Additional displays include the use of Splash Screen, which appears when the application is being loaded.

5.2.2 Communication Manager

The Communication Manager constitutes the Server functionality within the project. The server used is Apache Tomcat 5.0. This module performs the task of receiving the Search request from the MIDlet and processing it to be sent over to the Agent Platform.

This functionality is provided by the use of Servlets. Servlets are Java classes used to extend the capabilities of servers that host applications accessed via a request-response programming model. The Servlet used here has to communicate with two programs,

- Submitting MIDlet
- Listening Aglet

Depending on which program is communicating with the Servlet at a particular instance, it decides on the subsequent action. In case of MIDlet, it receives the request and stores it locally. This is then passed on to the listening Aglet. In case of the Aglet communication, it receives the search results and stores it to be sent back to the User Interface.

5.2.3 Agent Gateway

The Aglet gateway executes agents with the Tahiti server. This server listens on the specific port to the outgoing responses from the Servlet. The Aglet platform performs the task of dispatching itself to the Remote host and performing the Database Search Operation.

The Agent is a piece of code that can transport itself over the network, to a remote host to perform a specific task. Since the Aglet resides local to the Database, the search is less time consuming. The Tahiti server implements an agent protocol called ATP (Agent Transfer Protocol). This allows each system to be designated in terms of the URL as,

atp://< System Name>: <Port Number>

The Aglet dispatches itself to the ATP URL of the remote host where the database resides. This code transportation allows the Remote host to perform the Search more cost-effectively. The Aglet connects to the Local database through the JDBC Driver. The Database query changes depending upon the request sent from the User. The results from the database are retrieved in a Record Set Object before being processed. They are then retrieved in the appropriate form to be sent to the Server.

Once the search has been completed, the Aglet returns to the Agent Gateway. The Aglet then establishes the URL connection with the Servlet. Using this connection, the search results are sent back to the Servlet. The availability of the results is intimated to the Mobile Device through the form of an SMS (Short Messaging Service). This messaging is initiated from the Aglet through the Socket Connection.

5.2.4 Messaging Support

This module interacts with both the Aglet as well as the Mobile Device. The module is of high importance as it performs the task of intimating the User of the Result Availability.

The Aglet searches the Database for the request submitted and returns back to the Agent Gateway with the results. This is then submitted to the Servlet through the URL connection. The availability of the results is indicated to the Mobile Device through an SMS. The Aglet initiates this action through the Messaging Support Module. When the results are submitted to the Servlet, the Aglet initiates messaging in this module. This in turn sends the message to the Mobile Device. The User can then download the Results from the Servlet, by selecting the Download option in the User Interface.

5.2.5 Remote Host Database

The Database in the Remote Host is in Microsoft Access 2000. The entire data is split into four tables. Each of the tables has a significant relationship with one another. This makes the necessary data available to the user with the use of simple SQL queries.

The creation of the database involved creation of the tables, establishing relationships and finally collection and addition of relevant data.

6. SYSTEM TESTING AND IMPLEMENTATION

The System Testing deals with the process of testing the system as a whole. This is done after the integration process. The entire system is tested by verifying each module from top to bottom. The verification and validation process are being carried out. The errors that occur at the testing phase are eliminated and a well functioning system is developed.

6.1 TESTING METHODS

6.1.1 Unit Testing:

It focuses verification effort on the smallest unit of software design, the Module. The modules are tested separately during programming stage itself.

Each and every module is tested separately to check if its intended functionality is met. Some unit testing performed are,

- Checking the proper working of the system information and process list retrieval modules in the User Interface module
- Checking for proper connectivity between Mobile and Server and between the Server and Agent
- Ensuring that the Messaging support module functions independently without interfering with other functions.

Verifying the integrity of the MIDlet interface and ensuring correctness of command transmission

6.1.2Validation Testing

Validation testing can be defined in many ways but a simple definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the users.

After a validation test has been conducted one of the two possible conditions exists:

- The function or the performance characteristics confirm to the specification and are accepted
- Deviation from the specification is uncovered and the deficiency list is created

6.1.3 Output Testing

After performing the validation testing the next step is output testing of the proposed system since no system is useful if it does not produce the required output in the specific format. The outputs generated are displayed by the system under consideration are tested by asking the users about the formats required by them.

6.1.4 Integration Testing

It is the testing performed to detect errors on interconnection between modules. Here, all the modules pertaining to the Mobile, Server and Agent are combined to form the integrated applications. This is tested to ensure that they work in synchronization and without interference from one another.

7. CONCLUSION

The Project "Mobile Agents in Wireless Devices" uses an Agent - based structure to reduce Network load and latency. The project demonstrates these features through the use of a Book Search Application. The application entailed the use of the Aglet Workbench to Create and Dispatch Agents to the Remote host Database. The agent achieves this objective locally at a distant host. Thus it significantly reduces the connection time of the User by notifying the result availability through an SMS. It further enhances service functionality by operating without constant user input.

8. APPENDIX

APPENDIX A - SAMPLE CODE

SearchMid.java – MIDlet File

```
// MIDlet File
import javax.microedition.midlet.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import java.io.*;
import java.lang.Object;
import java.util.*;
public class SearchMid extends MIDlet implements
CommandListener,Runnable {

    private Form mainForm;
    private static Display display;
    private List mainList;
    private List Options;
    private final static Command CMD_EXIT = new Command("Exit",
Command.EXIT, 1);
    private final static Command CMD_BACK = new Command("Back",
Command.BACK, 1);
    private final static Command CMD_OK = new Command("ok",
Command.OK, 1);
    private TextField Bname;
    private String Binfo = new String();
    private String Breq = new String();
    private String AN = new String();
private String reply;
private String Tmp=new String();
Datagram dg;
byte[] bytes;
 /** user interface alert component. */
Alert splashScreenAlert;
// Wait for 2sec
static final int DefaultTimeout = 2000;
Image   splashScreen;
boolean imageLoaded;
// The current command processing thread.
private Thread commandThread;
// The URL to GET from the network.
private String URL;
     // URL Method.
     private String U_Method;
     //Final output screen box
     TextBox QResponse;
 public SearchMid() {


   display = Display.getDisplay(this);
     QResponse = new TextBox("SEARCH BOOK","Data sent for
     processing please wait...",100, TextField.ANY);
   Bname= new TextField("ENTER TEXT :", "", 50, TextField.ANY);
  try {
      splashScreen =
        Image.createImage("/pic/sp.png");
      imageLoaded = true;
     } catch (java.io.IOException ex) {  }
```

```
    splashScreenAlert = new Alert("Welcome to Book Search", "",
                    splashScreen, AlertType.INFO);
    splashScreenAlert.setTimeout(DefaultTimeout);
}

public void startApp() {

      String[] stringArray1 = {
              "New Search",
              "Download Results",
                  };
      mainForm = new Form("Text Field");
      mainForm.addCommand(CMD_BACK);
      mainForm.addCommand(CMD_OK);
      mainForm.setCommandListener(this);
      Image[] imageArray = null;
          try {
          // load the duke image to place in the image array
          Image icon = Image.createImage("/pic/Icon.png");
            // these are the images and strings for the choices.
          imageArray = new Image[] {
            icon,
            icon,
            icon,
            icon
          };
        } catch (java.io.IOException err) {
          // ignore the image loading failure the application can recover. }
        Options = new List("Options", Choice.EXCLUSIVE,
stringArray1,null);
        Options.addCommand(CMD_EXIT);
        Options.addCommand(CMD_OK);
        Options.setCommandListener(this);

         String[] stringArray = {
            "Title",
            "Author",
            "Publisher",
        "ISBN"      };
        mainList = new List("Search Book By", Choice.IMPLICIT,
stringArray,imageArray);
        mainList.addCommand(CMD_BACK);
        mainList.setCommandListener(this);
        mainForm.append(Bname);
        mainForm.setCommandListener(this);
        display.setCurrent(splashScreenAlert,Options);
    }

   // Invoked when Application is Paused
    public void pauseApp() {
    }

    // Invoked when Application is Destroyed
    public void destroyApp(boolean unconditional) {
    }
```

```
public void commandAction(Command c, Displayable d)
{
 if (d.equals(Options))
      {
     if (c == CMD_OK)
         {
          this.Breq = Options.getString(Options.getSelectedIndex());
              if("New Search".equals(Breq))
                {
                      Breq="Search";
                      display.setCurrent(mainList);
                }
              else if("Download Results".equals(Breq))
                {
                   Breq="Download";
                    try{
                      //URL to which data are to be directed
                      URL="http://localhost:8080/servlets-
examples/servlet/ByTitle_soc?Btype=J2ME&Breq=Download";
                      //URL Method
                      U_Method = "GET";
                      Start_Tread();
                      QResponse.addCommand(CMD_BACK);
                      QResponse.setCommandListener(this);
                      display.setCurrent(QResponse);
                       }catch(Exception e){ }
                }
        } else if (c == CMD_EXIT)
              {
                Bname.setString("");
            display.setCurrent(mainList);
            }
    }
      else if (d.equals(mainList))
            {
            // in the main list
             if (c == List.SELECT_COMMAND)
                    {
             this.Binfo = mainList.getString(mainList.getSelectedIndex());
                    display.setCurrent(mainForm);
             } else if (c == CMD_BACK)
                    {
             display.setCurrent(Options);
             }
         }
         else if (d.equals(mainForm))
            {
            if (c == CMD_OK)
                    {
                    try{
                    this.AN=Bname.getString();
                    StringBuffer b=new StringBuffer();
                    int i;
                    for(i=0;i<AN.length();i++)
                            {
                            if(AN.charAt(i)==' ')
                            {
                            b.append("%20");          }
                    else
                    b.append(AN.charAt(i));
                    }
                    String C22=""+b;
                String textstr ="Data sent for processing please wait...";
                QResponse.setMaxSize(textstr.length());
                QResponse.setString(textstr);
                //URL to which data are to be directed
                URL="http://localhost:8080/servlets-
examples/servlet/ByTitle_soc?Btype=J2ME&Bname="+C22+"&Binfo="
+Binfo+"&Breq="+Breq;
                //URL Method
                U_Method = "GET";
                Start_Tread();
                QResponse.addCommand(CMD_BACK);
                QResponse.setCommandListener(this);
                display.setCurrent(QResponse);
                }catch(Exception e){}
            } else if (c == CMD_BACK)
                {
                Bname.setString("");
            display.setCurrent(mainList);
             }
          } else if (d.equals(QResponse))
           {
           if (c == CMD_BACK)
                {
                Bname.setString("");
           display.setCurrent(Options);
```

```
        }
    }
       if (c == CMD_EXIT)
       {
        destroyApp(false);
        notifyDestroyed();
          }
   }

//Function invoked when threads are required.
       public void Start_Tread(){
              //Thread starts here
              synchronized (this) {
              if (commandThread != null) {
                     return; // process only one command at a time
                 }
        commandThread = new Thread(this);
        commandThread.start();
      }
    }

 //Funtion invoked when new thread is created
 public void run() {
              readContents(U_Method);
              readContents1();
        synchronized (this) {
        // signal that another command can be processed
        commandThread = null;
      }   }
```

```
private void readContents1()
{
String reply=new String();
        try{
        DatagramConnection                              dc=(DatagramConnection)
Connector.open("datagram://localhost:4444");
        bytes=new byte[1000];
        String req;
        req=new String("sdsa");
        bytes=req.getBytes();
        dg=dc.newDatagram(bytes,bytes.length);
        dc.send(dg);
        dg=dc.newDatagram(1000);
        dc.receive(dg);
        dc.close();
        reply=new String(dg.getData(),0,dg.getLength());
        }
        catch(Exception e){}
        QResponse.setMaxSize(reply.length());
        QResponse.setString(reply);
}

  // Read the content of the page for both HTTP request (GET or POST)
  private void readContents(String request) {
            StringBuffer buff = new StringBuffer();
            HttpConnection conn = null;
            OutputStream os = null;
            InputStream is = null;
            long len = -1;      // length of input stream
```

```
int ch=0;      //Character read through Input stream in ascii.
try{
    // URL is opened and typecasted to HttpConnection
     conn = (HttpConnection)Connector.open(URL);
        // Set the request method as POST or GET
conn.setRequestMethod(request);
                //open the input stream.
is = conn.openInputStream();
        //getting the length of theinut stream
            len=((HttpConnection)conn).getLength();
// If Content is not empty
if (len != -1) {
        // Read exactly Content-Length bytes
        for (int i = 0; i < len; i++) {
           if ((ch = is.read()) != -1) {
                    buff.append((char)ch);
        }
}
  }
 else {
  if (len==-1){
        len=100;
        }
  byte data[] = new byte[(int)len];
      int n = is.read(data, 0, data.length);
  for (int i = 0; i < n; i++) {
        ch = data[i] & 0x000000ff;
        buff.append((char)ch);
        }      }
```

```
            Tmp = ""+buff;
        QResponse.setMaxSize(Tmp.length());
        QResponse.setString(Tmp);
        if("Download".equals(Breq))
        {
        if("Title".equals(Binfo))
        title();
        else if("Author".equals(Binfo))
        author();
        else if("Publisher".equals(Binfo))
        pub();
        else if("ISBN".equals(Binfo))
        bookisbn();
        }
    }
        catch(IOException ioe){}
        //Closing the connections
            if (conn != null) {
                try {
                    is.close();
                    os.close();
                }
                catch (Exception ce) {}
            }
        }
private void title()
{       try
        {
        String t=new String();
```

```
StringBuffer isbn=new StringBuffer();
int i,j;
for(i=0;i<(Tmp.length()-1);i++)
{
if(Tmp.charAt(i)=='!')
{
        isbn.append("TITLE:");
        for(j=i+1;(Tmp.charAt(j)!='@');j++)
        {
        isbn.append(Tmp.charAt(j));
        }
        isbn.append("\n");
}
if(Tmp.charAt(i)=='@')
{
        isbn.append("ISBN:");
        for(j=i+1;(Tmp.charAt(j)!='*');j++)
        {
                isbn.append(Tmp.charAt(j));
        }
        isbn.append("\n");
}
if(Tmp.charAt(i)=='*')
{
        isbn.append("PRICE:");
        for(j=i+1;(Tmp.charAt(j)!='!');j++)
        {
                isbn.append(Tmp.charAt(j));
        }
```

```
                isbn.append("\n\n");
            }
        }
        t=""+isbn;
        QResponse.setMaxSize(t.length());
        QResponse.setString(t);
        }catch(Exception e){  }
    }

    private void author ()
    {
    // Contains printing format for Searching By Author Name
    }

    private void pub ()
    {
    // Contains printing format for Searching By Publisher Name
    }

    private void bookisbn ()
    {
    // Contains printing format for Searching By ISBN
    }

    }
```

ServletAglet.java – Aglet File

```
// Aglet File
import com.ibm.aglet.*;
import com.ibm.aglet.event.*;
import com.ibm.aglet.util.*;
import com.ibm.agletx.util.SimpleItinerary;
import java.lang.InterruptedException;
import java.io.Externalizable;
import java.io.ObjectInput;
import java.io.ObjectOutput;
import java.io.IOException;
import java.io.*;
import java.sql.*;
import java.util.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Enumeration;

public class ServletAglet extends Aglet {
        StringBuffer buf=new StringBuffer();
        String home = null;
        SimpleItinerary itinerary = null;
        String Bname=new String();
        String Binfo=new String();
        String C1=new String();
        String C2=new String();
        String C3=new String();
        String ser=new String();
        String C4=new String();
        static DatagramSocket sock;
        static DatagramPacket p;
        static String str,addr;
        String destination=null;
        public void onCreation(Object init)
        {
        itinerary = new SimpleItinerary(this);
        home = getAgletContext().getHostingURL().toString();
        try
        {
        URL        hp=        new        URL("http://90.0.1.26:8080/servlets-
examples/servlet/ByTitle_soc?Btype=AGENT&Bsel=name");
        URLConnection hpcon =hp.openConnection();
        DataInputStream                                          in=new
DataInputStream(hpcon.getInputStream());
        ser=in.readLine();
        StringTokenizer st=new StringTokenizer(ser,"*");
        while(st.hasMoreTokens())
        {
        Binfo=st.nextToken();
        Bname=st.nextToken();
        }
}catch(Exception e){}
        setText("Reading the Search value fro the servlet,.....");
        System.out.println(Bname+Binfo);
        waitMessage(10000);
        startTrip();
        }
        public void db()
        {
                setText("Searching database,.....");
                waitMessage(10000);
                if("Title".equals(Binfo))
                title();
                else if("Author".equals(Binfo))
                author();
                else if("ISBN".equals(Binfo))
                isbn();
                else if("Publisher".equals(Binfo))
                pub();
        }

public void title()
{

        // JDBC driver
        String driverName = "sun.jdbc.odbc.JdbcOdbcDriver";
        // JDBC connection URL
        String connectionURL = "jdbc:odbc:bsearch";
        Connection conn = null;
        Statement stmt = null;
        String sqlStatement = null;
        ResultSet rs = null;
        ResultSetMetaData rsmd = null;
        int rowCount = 0; //To store no of rows
        try{

        Class.forName(driverName);
        conn = DriverManager.getConnection(connectionURL, "",
"");
        stmt = conn.createStatement();
        sqlStatement = "SELECT titles.isbn as a,titles.title as
b,titles.price as c from titles where titles.title like '" + Bname + "%'";
        rs = stmt.executeQuery(sqlStatement);
        rsmd = rs.getMetaData();
        int i;
        StringBuffer b;
        while (rs.next())
        {
            rowCount++;
            C1 = rs.getString("a");
            C2 = rs.getString("b");
            b=new StringBuffer();
            for(i=0;i<C2.length();i++)
            {
            if(C2.charAt(i)==' ')
            {
            b.append("%20");
            }
            else
            b.append(C2.charAt(i));
            }
            String C22=""+b;
            C3 = rs.getString("c");
```

```
                    buf.append("!"+C22+"@"+C1+"*"+C3);
                }
                buf.append("!");
                rs.close();
                stmt.close();
                conn.close();
        }
            catch (Exception e)
            {
            e.printStackTrace();
            }
    }
    public void author()
    {
    // Performs Database Search operation with the Author key
    }
    public void pub()
    {
    // Performs Database Search operation with the Publisher key

    }
    public void isbn()
    {
    // Performs Database Search operation with the ISBN key

    }
            public synchronized void startTrip()
            {
            // get the address for trip
        try{
                destination = "atp://its251:4434";
                itinerary.go(destination, "sayHello");
            }
        catch(Exception e){ }
        }

        public boolean handleMessage(Message msg) {
                if (msg.sameKind("atHome")) {
                        atHome(msg);
                } else if (msg.sameKind("sayHello")) {
                        sayHello(msg);
                } else {
                        return false;
                }
                return true;
        }

        public void sayHello(Message msg) {
                // try back home
                try {
                        db();
                        setText("I'll go back to.. " + home);
                        waitMessage(1000);              // 1 second
                        // Go back home and Send "atHome" message to
owner this
                        itinerary.go(home, "atHome");
                } catch (Exception ex) {
                        ex.printStackTrace();
            }
        }

        public void atHome(Message msg) {
            setText("I'm back.");                // greetings
            String Result =buf.toString();

            try
            {
            int c;
            Socket s= new Socket("90.0.1.26",1234);
            InputStream in=s.getInputStream();
            OutputStream out=s.getOutputStream();
            String  str=new  String("Results  are  available.  Please
    download it!!");
            byte Buf[]=str.getBytes();
            out.write(Buf);
            out.flush();
            s.close();
            }catch(Exception e){ }
        try
        {
        URL      hp=      new      URL("http://90.0.1.26:8080/servlets-
    examples/servlet/ByTitle_soc?Btype=AGENT&Bsel=result&Bresult="+
    Result);
        URLConnection hpcon =hp.openConnection();
        DataInputStream                                        in=new
    DataInputStream(hpcon.getInputStream());
        String Bname11=in.readLine();
        System.out.println(Bname11);
        }catch(Exception e){}
        //waitMessage(5 * 1000);        // show message, 2 seconds
        dispose();                                  // dispose it self
      }
}
```

Bytitle_soc.java – Servlet

```
//Servlet File
import java.sql.*;
import java.util.*;
import java.io.*;
import javax.servlet.http.*;
import javax.servlet.*;
import java.net.*;
import java.lang.*;
public class ByTitle_soc extends HttpServlet
{
String Btype=new String();
String Breq=null;
String Bname=null;
String Bresult=new String();
String Bsel=null;
String Binfo=new String();
String abc1;
public void doGet(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException
{
```

```
res.setContentType("text/html");
PrintWriter out = res.getWriter();

//Get value of Query String Variable
Btype=req.getParameter("Btype");

//To process request from J2ME user
if("J2ME".equals(Btype))
{
        this.Breq=req.getParameter("Breq");
        if("Search".equals(Breq))
        {
        this.Bname=req.getParameter("Bname");
        this.Binfo=req.getParameter("Binfo");
        out.write("Your Request is Accepted. Thank You");

        }
        else if("Download".equals(Breq))
        {
        if("".equals(this.Bresult))
        out.write("Please Retry after some time.Network Busy!!!!");
        else
        out.write(this.Bresult);

        }

//To submit/receive data to/from Agent
} else if("AGENT".equals(Btype))
```

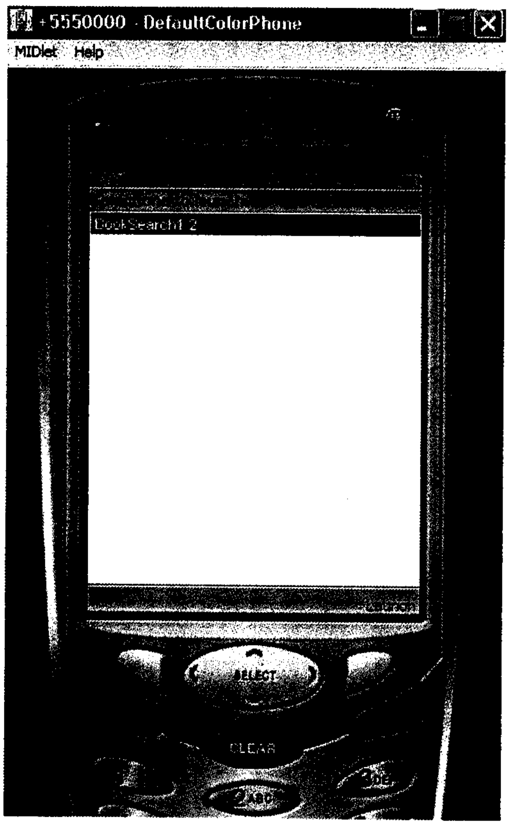
```
        {
            Bsel=req.getParameter("Bsel");
            if("name".equals(Bsel))
            {
            out.write(this.Binfo+"*"+this.Bname);
            }
            else if("result".equals(Bsel))
            {
            this.Bresult=req.getParameter("Bresult");
            out.write(Bresult);
            }

        } else
        out.write("Sorry Your Request cannot be processed now. Please try
after some time");
    }

}
```

Worker.java – Messaging Support

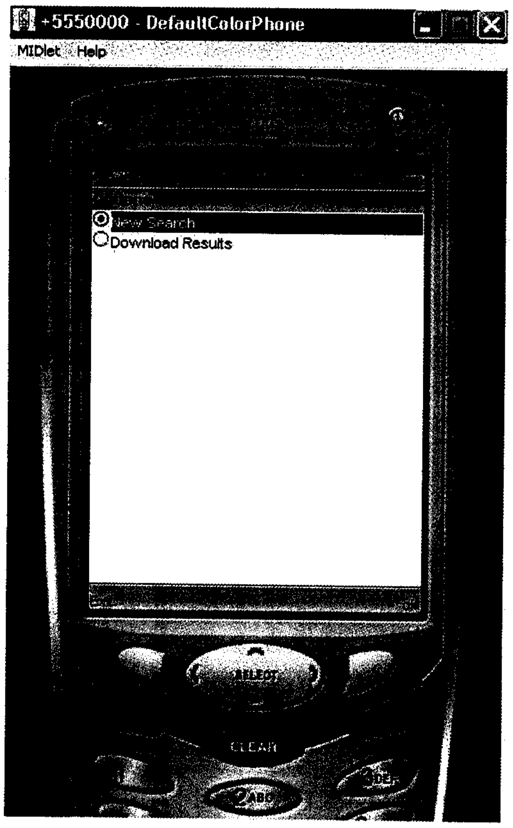
```
import java.io.*;
import java.net.*;
import java.util.*;
import java.lang.*;

public class Worker
{
static DatagramSocket sock;
static DatagramPacket p;
static String str,head,req,uname,pwd,addr;
static boolean b,reply;

public static void main(String a[]) throws Exception
 {
        sock=new DatagramSocket(4444);
        Date d=new Date();
        p=new DatagramPacket(new byte[1000],1000);
        sock.receive(p);
        addr=p.getAddress().toString();
        str=new String(p.getData(),0,p.getLength());
        long j2meTime=d.getTime();
        System.out.print(d.getTime());
        StringBuffer buf=new StringBuffer();
        int c;
        ServerSocket sc= new ServerSocket(1234);
        Socket s= sc.accept();
        InputStream in=s.getInputStream();
        OutputStream out=s.getOutputStream();
        while((c=in.read())!=-1)
        {
                buf.append((char)c);
        }
        s.close();
        String reply = ""+buf;
        System.out.println(reply);
        byte[] bytes=reply.getBytes();
        p=new
DatagramPacket(bytes,bytes.length,p.getAddress(),p.getPort());
        sock.send(p);
        long recTime=System.currentTimeMillis();
        System.out.println(System.currentTimeMillis());
        long diff=((recTime-j2meTime)/1000);
        System.out.println("The time difference" + diff  + " seconds");
    }
}
```

APPENDIX B - SCREEN SHOTS

APPLICATION STARTUP

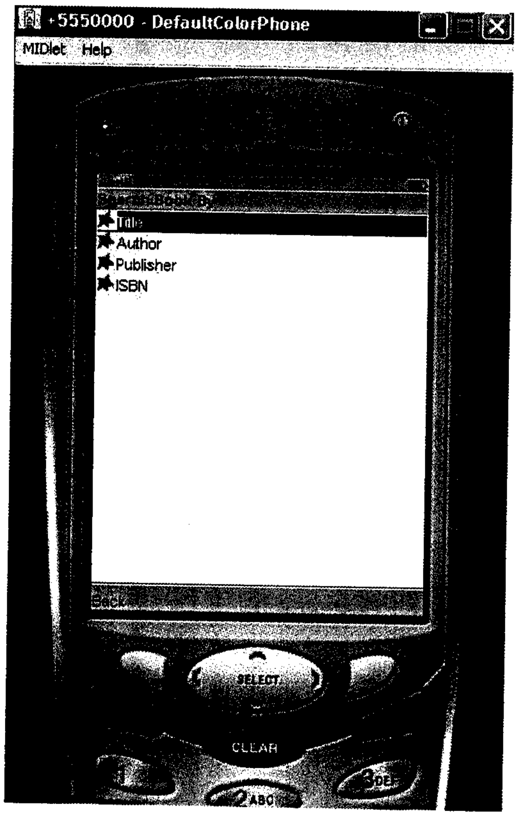


USER OPTIONS

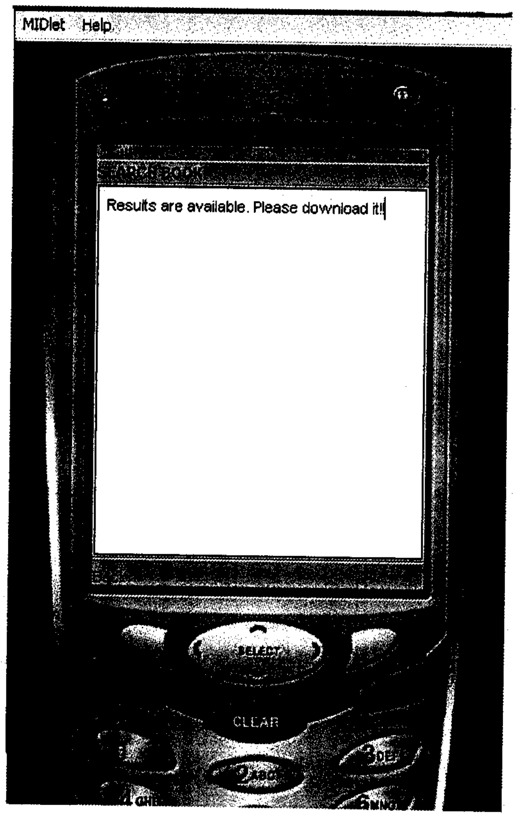


New Search [✓]
Download Results

SEARCH OPTIONS

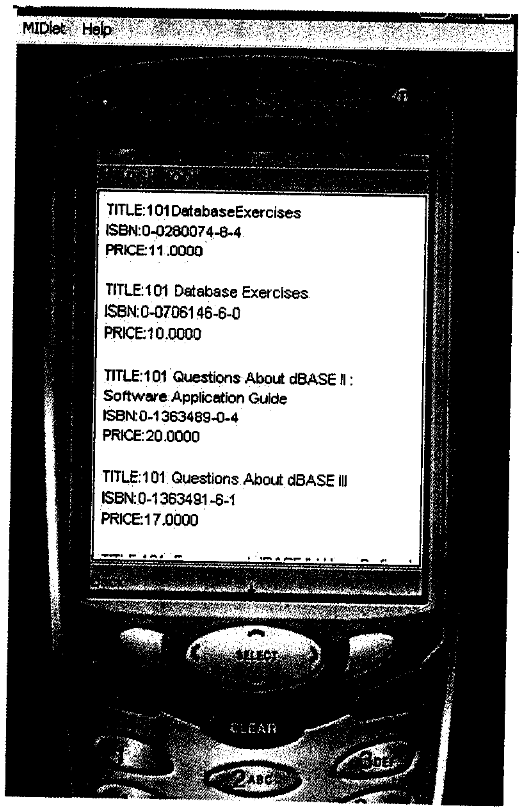


SMS RECEPTION



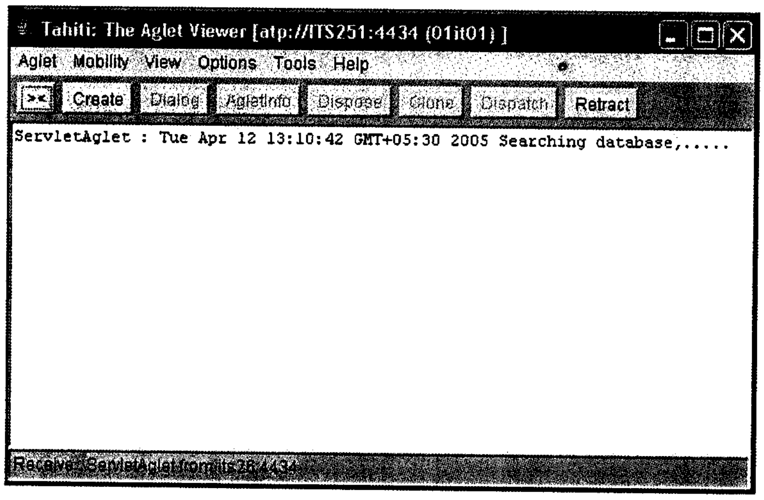
Results are available. Please download it!!

RESULT DISPLAY FOR - SEARCH BY TITLE



```
TITLE:101DatabaseExercises
ISBN:0-0280074-8-4
PRICE:11.0000

TITLE:101 Database Exercises
ISBN:0-0706146-6-0
PRICE:10.0000

TITLE:101 Questions About dBASE II :
Software Application Guide
ISBN:0-1363489-0-4
PRICE:20.0000

TITLE:101 Questions About dBASE III
ISBN:0-1363491-6-1
PRICE:17.0000
```

AGLETS – TAHITI GUI


Tahiti: The Aglet Viewer [atp://ITS251:4434 (01it01)]
Aglet Mobility View Options Tools Help
Create
Dialog
AgletInfo
Dispose
Clone
Dispatch
Retract

```
ServletAglet : Tue Apr 12 13:10:42 GMT+05:30 2005 Searching database,.....
```

Receive : ServletAglet from its28:4434

REFERENCES

- Bill Venners , 'Solve real problems with aglets, a type of mobile agent' in **JavaWorld**
- Bill Venners, 'Under the Hood: The architecture of aglets' in **JavaWorld**
- Bret Sommers, 'Agents: Not just for Bond anymore' in **JavaWorld**
- Danny Lange and Mitsuru Oshima, 'Programming and Deploying Java™ Mobile Agents with Aglets™', Addison Wesley, 1998, (ISBN: 0-201-32582-9)
- George Lawton , 'Agents to roam the Internet' in **SunWorld Online**
- http://www.javaworld.com/javaworld/jw-04-1997/jw-04-agents.html
- http://www.javaworld.com/javaworld/jw-04-1997/jw-04-servlet.html
- http://www.sun.java.com
- http://www.trl.ibm.co.jp/aglets
- Jonarthan Knudson, 'Wireless Java Programming with J2ME 2.0'