Artificial Intelligence Techniques & Agent Technology

E-Library system

(Multi-agent)

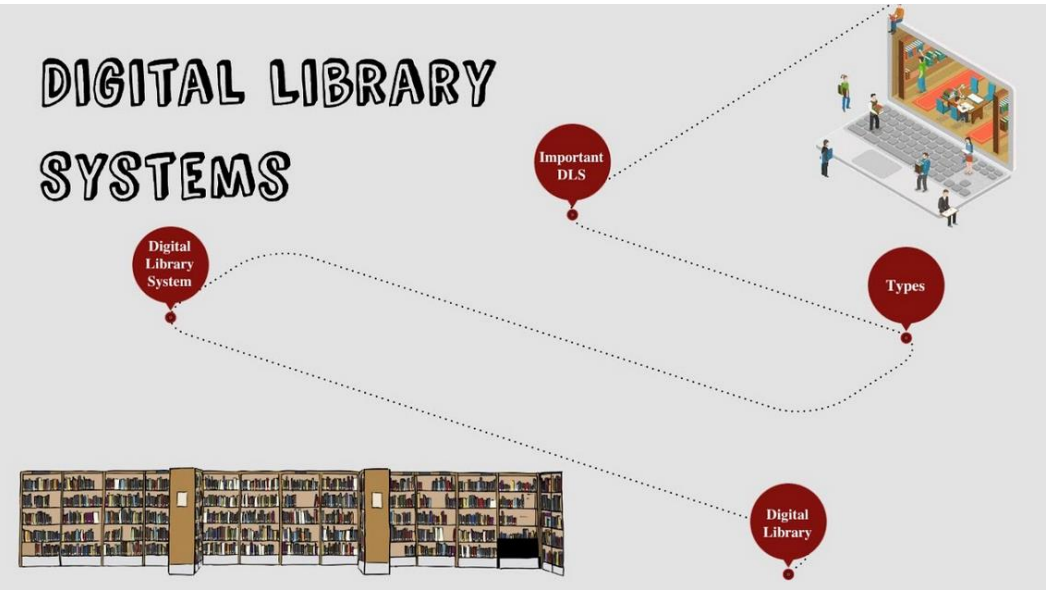
TABLE OF CONTENT

1. Introduction

Multi-agent systems (MASs) are a new and promising area in the field of distributed artificial intelligence (DAI), as well as in mainstream computer science. These systems are compounds of relatively autonomous and intelligent parts, called agents. The E-Library system is designed using multi-agent technology.

Multi-specialist frameworks have been significantly adding to the advancement of the hypothesis and the act of mind-boggling appropriated frameworks and, specifically, they have shown the possibility to address basic issues in high velocity, crucial, content-rich, and dispersed data applications where shared interdependencies, dynamic conditions, vulnerability, and complex control assume a part.

Previous online library management system (LMS) acts as a tool to search books online and purchase or read books. The e-library multi agent system supports the e-libraries to encounter all the issues concurrently. The users don't wait in a queue for a long period to find a version of books from the e-library. The single system contains all the data's in it. The users have to assess the system and provide an entry in it. Through the system the users can find the books with versions in the bookshelves. The system is designed with the basic features such as users can purchase, search books with latest versions.



1.1 AIM

Decision-making is a core topic of research in the fields of artificial intelligence and machine learning. Using these past e-library systems users can't find the latest version of the books. In previos e-library systems users can only find a book. If the user needs the latest version of the book he/she must refer to the internet to find that.

Build an e-library system using multi-agent technology.

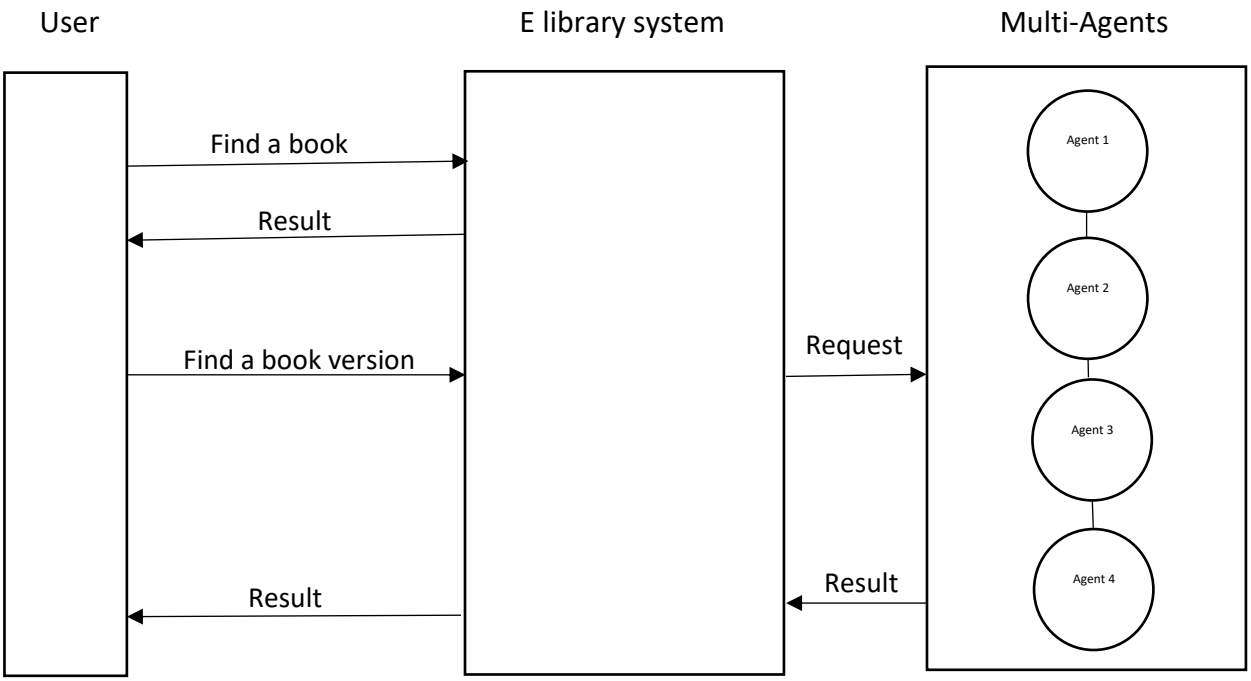
The aim is:
• Increased efficiency of access to information from manuals, and reduce
average
time for searching for the latest version of a book.
• Quality maintenance.
• The latest version of the books is more valuable for every field.
• Find a target book and updated book version indexing all the connected e-
libraries.

1.2 OBJECTIVES

- The main objective is to find a book with the latest version of the book efficiency. The user's time and effort are spent searching on the internet to find the latest book version.

- The system displays the processed information and file to download.

2.DESIGN AND IMPLEMENTATIONS



2.1 IMPLEMENTATIONS

To implement this E-Library System Mini project I used JADE as Multi Agent Development Environment. According to the FIPA guidelines for interoperable multi-agent systems, JADE (Java Agent Development Framework) is a software environment for developing agent systems for the administration of networked information resources. JADE framework the basic application will be handle the major core incident that happen on the E-Library system.

2.2 Create an agent using java

Most of the time, a java agent is just a jar file made just for it. To modify the bytecode that has already been loaded into the JVM, it makes use of the Instrumentation API of the JVM. We should characterize two procedures for a specialist to work.

This mini project is based on the scenario that you have answered in TMA3. Based on the scenario that you selected, implement the solution you proposed using any MAS framework.

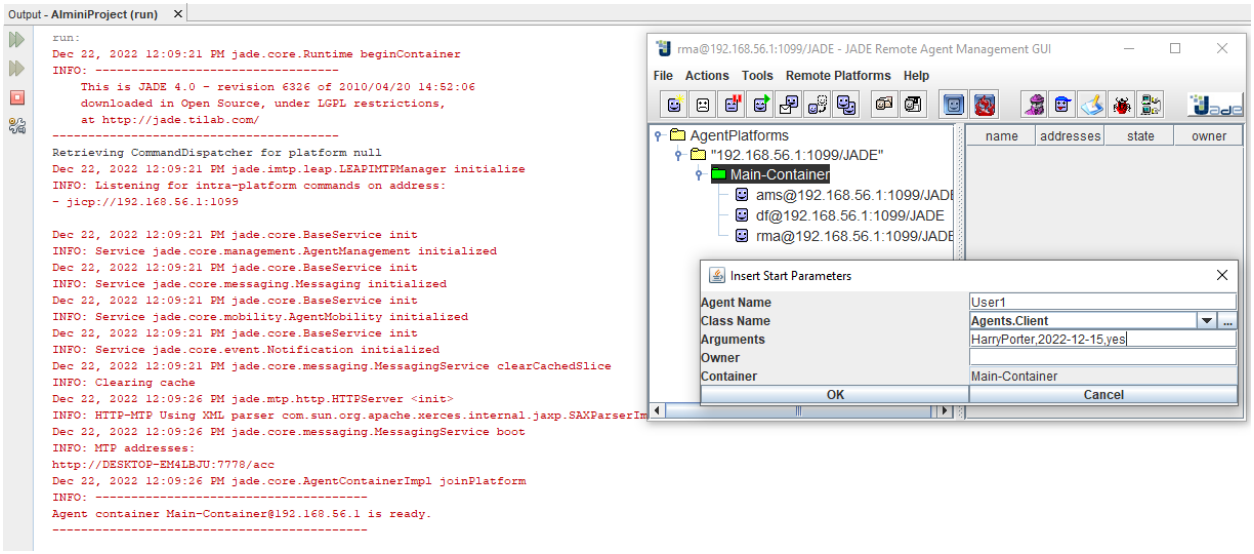
The design solution is implemented with the JADE framework and the basic application will be handle the major core incident that happen on the E-Library system.

In the design solution has 3 major categories;

- Client Agent
- Main library Agent
- Other library searching part

2.2.1 Client Agent

I design a simple book information related things and it communicate with the main Library system and other library systems as well.



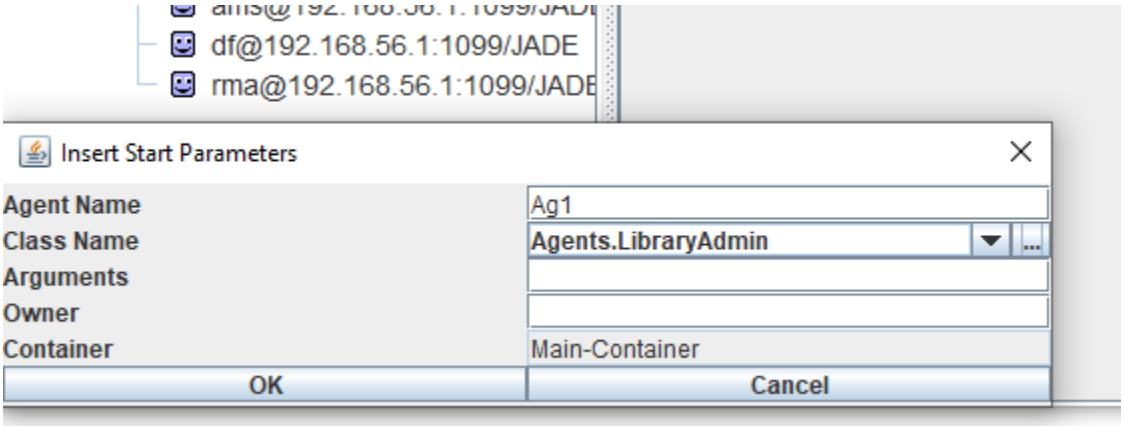
Agent name : String
Class name : Select agent class (Client)
Arguments : Select a book name, Current date, Do you need to find the latest book version? (yes or no)

2.2.2 Defined paramiters

```
http://DESKTOP-EM4LBJU:7778/acc
Dec 22, 2022 12:09:26 PM jade.core.AgentContainerImpl joinPlatform
INFO: ----------------------------------------
Agent container Main-Container@192.168.56.1 is ready.
--------------------------------------------
[*] Agent - Client User1@192.168.56.1:1099/JADE was created.
Appointment required - HarryPorter for date - 2022-12-15 (Book - yes)
```

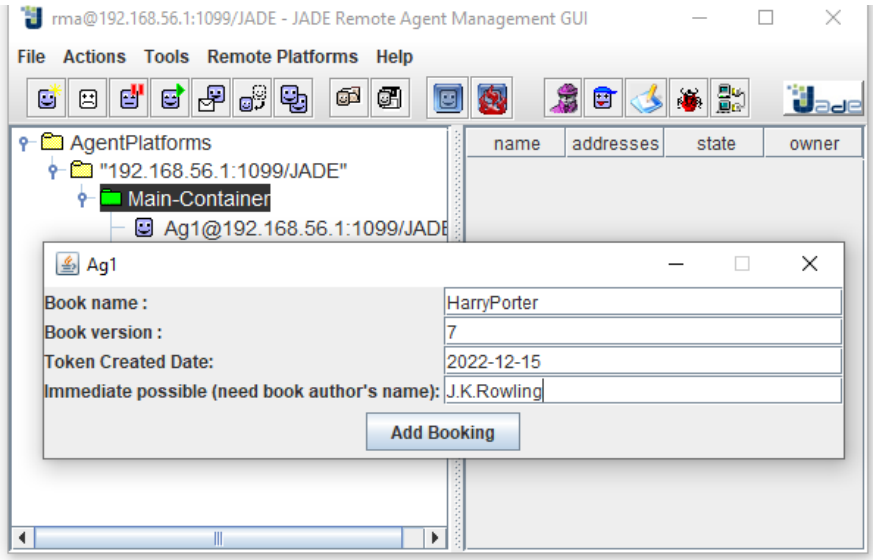
2.2.3 Main Library agent

Main library searching details




```
Trying to allocate HarryPorter
Book assign: (Appointment Type : HarryPorter )
No Results found
-----------
[*] Agent - Book Ag1@192.168.56.1:1099/JADE was created.
Trying to allocate HarryPorter
Book assign: (Appointment Type : HarryPorter )
No resulting value found
-----------
```

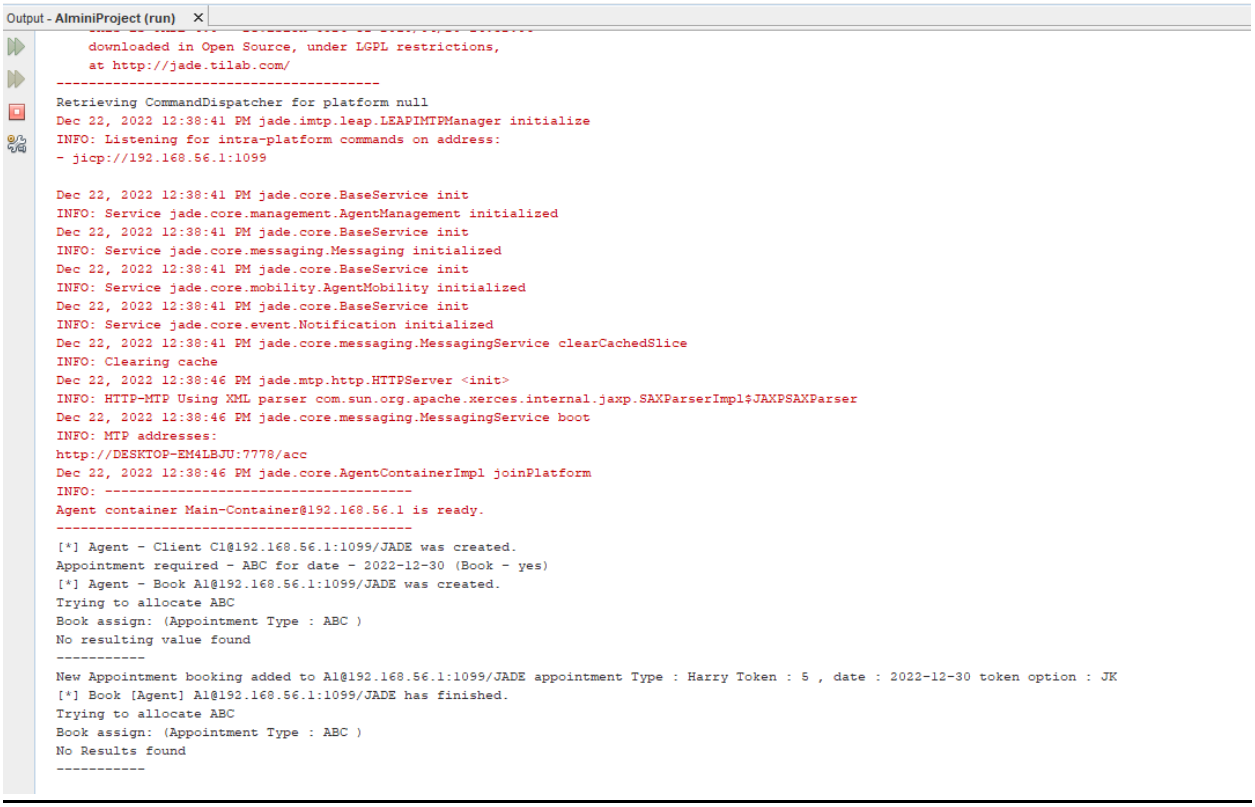
And we can enter another details to next form.



In the dialog, we can see the following information.

- Book name : String
- Book version : Interger
- Token created date : current date
- Book authors name : String

2.2.4 As soon as possible searching



3. APEENDIX

3.1 E-Library user interface

```
package Library;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import Agents.LibraryAdmin;
import javax.swing.JTextField;

public class LibraryInterface extends JFrame {

   private LibraryAdmin myAgent;
   private JTextField Appointmentype, token_Numb, token_option, token_issue_date;
```

```
  public LibraryInterface(LibraryAdmin a) {
    super(a.getLocalName());
    myAgent = a;

    JPanel p = new JPanel();
    p.setLayout(new GridLayout(4, 4));
    p.add(new JLabel("Book name : "));
    Appointmentype = new JTextField(15);
    p.add(Appointmentype);

    p.add(new JLabel("Book version : "));
    token_Numb = new JTextField(15);
    p.add(token_Numb);

    p.add(new JLabel("Token Created Date: "));
    token_issue_date = new JTextField(15);
    p.add(token_issue_date);

    p.add(new JLabel("Immediate possible (need book author's name): "));
    token_option = new JTextField(15);
    p.add(token_option);

    getContentPane().add(p, BorderLayout.CENTER);

    JButton addButton = new JButton("Add Booking");
    addButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent ev) {
        try {
          String appointment_Type = Appointmentype.getText().trim();
          String token_Number = token_Numb.getText().trim();
          String token_optin_value = token_option.getText().trim();
          String date = token_issue_date.getText().trim();

          myAgent.updateRecBook(appointment_Type, Integer.parseInt(token_Number),
token_optin_value, date);
          Appointmentype.setText("");
          token_Numb.setText("");
          token_option.setText("");
          token_issue_date.setText("");
          JOptionPane.showMessageDialog(LibraryInterface.this, "[*] Add Appointment Type for
booking!");
        } catch (Exception e) {
```

```
                JOptionPane.showMessageDialog(LibraryInterface.this, "Invalid value " + e.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
            }
          }
       });
       p = new JPanel();
       p.add(addButton);
       getContentPane().add(p, BorderLayout.SOUTH);

       // Make the agent terminate when the user closes
       // the GUI using the button on the upper right corner
       addWindowListener(new WindowAdapter() {
          public void windowClosing(WindowEvent e) {
             myAgent.doDelete();
          }
       });

       setResizable(false);
    }

    public void showGui() {
       pack();
       Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
       int centerX = (int) screenSize.getWidth() / 2;
       int centerY = (int) screenSize.getHeight() / 2;
       setLocation(centerX - getWidth() / 2, centerY - getHeight() / 2);
       super.setVisible(true);
    }

}
```

3.2 Token transfer

```
package Agents;

public class Tokensystem {

    private int slot_number;
    private String toekn_option;
    private String date;

    public Tokensystem(int slot_number, String toekn_option, String date) {
       this.slot_number = slot_number;
       this.toekn_option = toekn_option;
```

```
        this.date = date;
    }
    public int getSlot_number() {
        return slot_number;
    }
    public void setSlot_number(int slot_number) {
        this.slot_number = slot_number;
    }

    public String getToekn_option() {
        return toekn_option;
    }

    public void setToekn_option(String toekn_option) {
        this.toekn_option = toekn_option;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }
}
```

3.3 Library agent

```
package Agents;

import Library.LibraryInterface;
import jade.core.AID;
import jade.core.Agent;
import jade.core.behaviours.Behaviour;
import jade.core.behaviours.CyclicBehaviour;
import jade.core.behaviours.OneShotBehaviour;
import jade.core.behaviours.TickerBehaviour;
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.domain.FIPAException;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import java.util.Hashtable;
```

```
public class LibraryAdmin extends Agent {

   public static Hashtable appointmentRec;
   // The GUI by means of which the user can add books in the catalogue
   private LibraryInterface myGui;

   // Put agent initializations here
   protected void setup() {

      System.out.println("[*] Agent - Book " + getAID().getName() + " was created.");
      appointmentRec = new Hashtable();

      // Create and show the GUI
      myGui = new LibraryInterface(this);
      myGui.showGui();

      // Register the rec book info
      DFAgentDescription dfd = new DFAgentDescription();
      dfd.setName(getAID());
      ServiceDescription sd = new ServiceDescription();
      sd.setType("Appointment-book");
      sd.setName("Appointmen portfolio JADE");
      dfd.addServices(sd);
      try {
         DFService.register(this, dfd);
      } catch (FIPAException fe) {
         fe.printStackTrace();
      }


      addBehaviour(new OfferRequestsServer());


      addBehaviour(new BookingAppointmentServer());
   }

   // Put agent clean-up operations here
   protected void takeDown() {
      // Deregister from the yellow pages
      try {
         DFService.deregister(this);
      } catch (FIPAException fe) {
         fe.printStackTrace();
```

```
    }
    // Close the GUI
    myGui.dispose();
    // Printout a dismissal message
    System.out.println("[*] Book [Agent] " + getAID().getName() + " has finished.");
  }

  /**
   * This is invoked by the GUI when the user adds a new book for appointment
   * adding
   */
  public void updateRecBook(final String appointment_Type, final int solt_Number, final String
token_option, final String date) {
    addBehaviour(new OneShotBehaviour() {
      public void action() {
        appointmentRec.put(appointment_Type, new Tokensystem(new Integer(solt_Number),
token_option, date));
        System.out.println("New Appointment booking added to " + getAID().getName() + "
appointment Type : " + appointment_Type + " Token : "
            + solt_Number + " , date : " + date + " token option : " + token_option);
      }
    });
  }

  private class OfferRequestsServer extends CyclicBehaviour {

    public void action() {
      MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.CFP);
      ACLMessage msg = myAgent.receive(mt);
      if (msg != null) {
        // CFP Message received. Process it
        String appointmentType = msg.getContent();
        ACLMessage answer = msg.createReply();

        Tokensystem token = (Tokensystem) appointmentRec.get(appointmentType);
        if (token != null) {
          // The requested appointment is available for booking. Reply with the slot
          answer.setPerformative(ACLMessage.PROPOSE);
          answer.setContent(String.valueOf(token.getSlot_number()));
        } else {
          // The requested product is NOT available for booking.
          answer.setPerformative(ACLMessage.REFUSE);
          answer.setContent("Not Available");
        }
```

```
            myAgent.send(answer);
         } else {
            block();
         }
      }

   }  // End of inner class OfferRequestsServer

   private class BookingAppointmentServer extends CyclicBehaviour {

      public void action() {
         MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.ACCEPT_PROPOSAL);
         ACLMessage msg = myAgent.receive(mt);
         if (msg != null) {
            // ACCEPT_PROPOSAL Message received. Process it
            String appointmentType = msg.getContent();
            ACLMessage answer = msg.createReply();

            Tokensystem token = (Tokensystem) appointmentRec.remove(appointmentType);
            if (token != null) {
               answer.setPerformative(ACLMessage.INFORM);
               System.out.println("The Appointment " + appointmentType + " was assign to agent " +
msg.getSender().getName());
            } else {
               // The requested booking has been allocated
               answer.setPerformative(ACLMessage.FAILURE);
               answer.setContent("Not Available");
            }
            myAgent.send(answer);
         } else {
            block();
         }
      }

   }  // End of inner class OfferRequestsServer

}
```

3.4 Client agent

```
package Agents;

import static Agents.LibraryAdmin.appointmentRec;
import jade.core.AID;
import jade.core.Agent;
import jade.core.behaviours.Behaviour;
import jade.core.behaviours.TickerBehaviour;
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.domain.FIPAException;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;

public class Client extends Agent {

   // The name of the appointment
   private String allocateAppointment;
   private String allocateDate;
   private String LibraryAdmin;

   private AID[] AdminAgents;
   // Time to request product to agents
   private static final int REQUEST_TIME = 30000;

   private String AS_SOON = "YES";

   // Put agent initializations here
   protected void setup() {
      // Print a welcome message
      System.out.println("[*] Agent - Client " + getAID().getName() + " was created.");

      // Get the name of the appointment type as a start-up argument
      Object[] args = getArguments();
      if (args != null && args.length > 0) {
         allocateAppointment = (String) args[0];
         allocateDate = (String) args[1];
         try {
            if (args.length > 2) {
               String my_book = (String) args[2];
               if (my_book != null && !my_book.isEmpty()) {
```

```
                LibraryAdmin = my_book;
            }
          }
        } catch (Exception e) {
          System.out.println("[*] Couldn't Find Allocating Book's Name");
        }

        System.out.println("Appointment required - " + allocateAppointment + " for date - " +
allocateDate + " (Book - " + LibraryAdmin + ")");

        addBehaviour(new TickerBehaviour(this, REQUEST_TIME) {
          protected void onTick() {
            System.out.println("Trying to allocate " + allocateAppointment);
            // Update the list of seller agents
            DFAgentDescription agentDesc = new DFAgentDescription();
            ServiceDescription sd = new ServiceDescription();
            sd.setType("Appointment-book");
            agentDesc.addServices(sd);
            try {
              DFAgentDescription[] result = DFService.search(myAgent, agentDesc);
              System.out.println("Book assign: (Appointment Type : " + allocateAppointment + " )");
              AdminAgents = new AID[result.length];
              if (result.length > 0) {
                for (int i = 0; i < result.length; ++i) {
                  Tokensystem tracker = (Tokensystem) appointmentRec.get(allocateAppointment);
                  if (tracker != null) {
                    String my_date = tracker.getDate();
                    if (my_date.equalsIgnoreCase(allocateDate)) {
                      if (LibraryAdmin == null || LibraryAdmin.isEmpty()) {
                        AdminAgents[i] = result[i].getName();
                        System.out.println("User : " + AdminAgents[i]);
                        System.out.println("Found book Agent By :" + AdminAgents[i].getName());
                      } else {
                        System.out.println("Requested Admin : " + LibraryAdmin);
                        String found_book = ("" + result[i].getName()).split("@")[0];
                        System.out.println("Found book : " + found_book);
                        if (LibraryAdmin != null && LibraryAdmin.equalsIgnoreCase(found_book) &&
tracker.getToekn_option().equalsIgnoreCase(AS_SOON)) {
                          System.out.println(" Book found, But, not allowed to Immediate service");
                        } else {
                          AdminAgents[i] = result[i].getName();
                          System.out.println("User : " + AdminAgents[i]);
                          System.out.println("Found Book Agent By :" + AdminAgents[i].getName());
                        }
```

```
                        }

                    } else {
                        System.out.println("Appointment Type found, but different dates");
                    }
                } else {
                    System.out.println("No resulting value found");
                }

            }
        } else {
            System.out.println("No Results found");
        }
        System.out.println("-----------");
      } catch (FIPAException fe) {
        fe.printStackTrace();
      }

      // Perform the request
      myAgent.addBehaviour(new BookingRequest());
    }
  });
 } else {
  // Make the agent terminate
  System.out.println("The requested book is not available.");
  doDelete();
 }
}

// Put agent clean-up operations here
protected void takeDown() {
 // Printout a dismissal message
 System.out.println("Client Agent " + getAID().getName() + " has finished.");
}


private class BookingRequest extends Behaviour {

 private AID selectedDoctor;
 private String selectedSlot;
 private int repliesCnt = 0;
 private MessageTemplate mt;
 private int step = 0;
```

```
 public void action() {
        switch (step) {
           case 0:

              ACLMessage cfp = new ACLMessage(ACLMessage.CFP);
              for (int i = 0; i < AdminAgents.length; ++i) {
                 cfp.addReceiver(AdminAgents[i]);
              }
              cfp.setContent(allocateAppointment);
              cfp.setConversationId("Appointment-Commerce");
              cfp.setReplyWith("cfp " + System.currentTimeMillis()); // Unique value
              myAgent.send(cfp);
              // Prepare the template to get proposals
              mt = MessageTemplate.and(MessageTemplate.MatchConversationId("Appointment-
Commerce"), MessageTemplate.MatchInReplyTo(cfp.getReplyWith()));
              step = 1;
              break;
           case 1:

              ACLMessage reply = myAgent.receive(mt);
              if (reply != null) {
                 // Reply received
                 if (reply.getPerformative() == ACLMessage.PROPOSE) {
                    String selected_slot = reply.getContent();
                    if (selectedDoctor == null || !selected_slot.equalsIgnoreCase(selectedSlot)) {
                       // This is the best option at present
                       Tokensystem tracker = (Tokensystem) appointmentRec.get(allocateAppointment);
                       if (tracker != null) {
                          String my_date = tracker.getDate();
                          if (my_date.equalsIgnoreCase(allocateDate)) {
                             selectedSlot = selected_slot;
                             selectedDoctor = reply.getSender();
                             System.out.println("Define parameters : " +
reply.getAllUserDefinedParameters());
                          } else {
                             System.out.println("Appointment Type found, but different dates");
                          }
                       } else {
                          System.out.println("No resulting value found");
                       }

                    }
                 }
                 repliesCnt++;
```

```
            if (repliesCnt >= AdminAgents.length) {
               // We received all replies
               step = 2;
            }
         } else {
            block();
         }
         break;
      case 2:

         ACLMessage booking = new ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
         booking.addReceiver(selectedDoctor);
         booking.setContent(allocateAppointment);
         booking.setConversationId("Appointment-Commerce");
         booking.setReplyWith("Booking " + System.currentTimeMillis());
         myAgent.send(booking);

         mt = MessageTemplate.and(MessageTemplate.MatchConversationId("Appointment-
Commerce"), MessageTemplate.MatchInReplyTo(booking.getReplyWith()));
         step = 3;
         break;
      case 3:
         // Receive the booking reply
         reply = myAgent.receive(mt);
         if (reply != null) {
            // booking reply received
            if (reply.getPerformative() == ACLMessage.INFORM) {
               // allocating successful. We can terminate
               System.out.println(allocateAppointment + " has been allocated to agent " +
reply.getSender().getName());
               System.out.println("Slot = " + selectedSlot);
               myAgent.doDelete();
            } else {
               System.out.println("Error: Request slot already allocated");
            }
            step = 4;
         } else {
            block();
         }
         break;
      }
   }

   public boolean done() {
```

```
        if (step == 2 && selectedDoctor == null) {
           System.out.println("Error: " + allocateAppointment + " is not for booking.");
        }
        return ((step == 2 && selectedDoctor == null) || step == 4);
      }
   } // End of inner class RequestPerformer

}
```

REFERENCES

1.Frans.A.Oliehoek, Department of Computer Science, University of Liverpool, The MADP Toolbox: An Open Source Library for Planning and Learning in (Multi-)Agent Systems.

2.En.wikipedia.org. 2021. Java Agent Development Framework - Wikipedia. [online] Available at: https://en.wikipedia.org/wiki/Java_Agent_Development_Framework Accessed 16 November 2021.

3. Sites.owu.edu. 2021. Blue Jade Library – A Series of Series. [online] Available at: https://sites.owu.edu/seriesofseries/blue-jade-library/ Accessed 16 November 2021.

4. What-when-how.com. 2021. Agent Technology (information science). [online] Available at: <http://what-when-how.com/information-science-and-technology/agent-technology-information-science/> [Accessed 16 November 2021].