LOCATION BASED MULTI AGENT SOLUTION FOR DISTRIBUTED SERVICES

H.A.A Dhanushka Hapuarachchi

168283L

Degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa
Sri Lanka

March 2019

LOCATION BASED MULTI AGENT SOLUTION FOR DISTRIBUTED SERVICES

Hapu Arachchige Amal Dhanushka Hapuarachchi

168283L

Thesis submitted in partial fulfillment of the requirements for the degree Master of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa
Sri Lanka

March 2019

Declaration

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Name of the student: Signature:

H.A.A Dhanushka Hapuarachchi.

Date:

The above candidate has carried out research for the Master's thesis under my supervision.

Name of the supervisor: Signature:

Prof. A.S. Karunananda.

Date:

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. A.S. Karunananda for the continuous support of my MSC study and research, for his patience, motivation, enthusiasm, and immense knowledge. His expert guidance helped me in all the time of research allowing me to grow as a research scientist. This has been a period of intense learning for me, not only in the scientific arena, but also on a personal level.

My sincere thanks goes to all the lecturers of the Department of Computational Mathematics for their insightful comments and encouragement. Without their precious support it would not be possible to conduct this research.

Furthermore, I would like to convey thanks to my family members, fellow colleagues and friends for the support they have given me for successfully complete this research project.

Abstract

Attending an emergency service break down in fields such as water, electricity, fire and gas in a domestic environment to find a service provider has been an issue for a technically non competent person or a disable person. Even technically competent person sometime cannot take quick decisions in such emergency situations.

In day today life we always expect support or services from other people and people do businesses providing services. Services such as vehicle/ three wheel hiring, technician’s service of carpenter/plumber, goods transport, ambulance service, tuition classes and consultancy, we need services from other parties. The main thing is we can enjoy the service if it is provided by most suitable person on required time at required place. But most of the time we are unable to have satisfied service from correct service provider on required time due to not awareness about suitable service providers. Service providers like Uber, Pick Me provides this kind of services but limited to vehicle hires in Sri Lanka and there are lots of systems for taxi dispatch in the world, but it is hard to find common system for distributed services. Handling this kind of problems in dynamic, distributed and complex environment with involve uncertainty is a very crucial process. Multi-agent systems are in the domain of distributed artificial intelligence and it presents an alternative way of designing distributed control systems with autonomous and cooperative agents. Those agents there exhibit modularity, robustness, flexibility and adaptability. A research has been conducted to develop a multi agent based solution to find most suitable services and service providers even if client does not have knowledge about the area of domestic service or the issue. We have developed a Software for client to enter required parameters and request a particular service. In this situation service providers are registered in the system and distributed over the island and they are waiting for a request from a client. System has been tested while doing developments and we obtained encouraging results for leading formal evaluation.

***Keywords**. Agents, Complex systems, Multi-agent systems*

Table of Contents

Table of Figures

List of Appendices

Chapter 1

Introduction

1.1 Prolegomena

Nowadays, requesting a service such as hiring a taxi using a mobile device is a normal thing and technology has been developed up to such a scale [1]. AI technics such as Multi Agent technology has contributed a lot to this technological growth [2]. Those multi agent based systems always support people providing valuable services cost and time effectively. Most of the time in our lives we expect services from other people. It is true that without the support of other people we cannot survive. But in many case in our lives we meet problems suddenly and become helpless and cased to waste time and money. In domestic environment most of the people are un-aware about domestic systems or technologies and also there are disable people who know the technology but unable to resolve a problem in an emergency siltation. Sometime even technically competent person cannot take quick decisions or the taken decision might not be accurate in such emergency situations. In that case, if we can have support or service quicker as possible then we can save our valuable time and can be happy overcoming those issues.

Today providing a service is a business and also people earn lot of money even as a profession. As example if there is a water leak or pipe line damage in our bathroom, we expect a skilled plumber or technician quicker as possible to repair it. But sometime some people in domestic environment do not know, to whom they should call to solve this problem. In that situation those people become confused to find suitable person or a service to attend this mater. They go here and there and waste time and money to find a suitable service provider. Once we found someone, then the damage would be higher or massive. In this situation, it is required to have an intelligent system to assist those people. As another example, if your vehicle is broken down somewhere in remote area where you cannot access or reach to any workshop, how your vehicle is get repaired and complete your journey. In general what we do is, we ask help from people near here and there or call one of our friends and wait till he arrives. Most of the time we waste time and get tired finding a workshop which may be closer but we are not aware. Also in scenarios like vehicle/ three wheel hiring, technician's service of carpenter/plumber, goods transport (ex: Gas tanks), ambulance service, tuition classes and consultancy we expect service from other parties. The main thing is we can enjoy the service if it is provided by most suitable person on required time at required place. Though there are related systems like taxi dispatch systems in the world, It is hard to find this kind of common distributed service systems [2] [3]. Service providers like Uber, Pick Me provides this kind of services but limited to vehicle hires in Sri Lanka. Therefore, developing such intelligent system for managing this kind of scenarios is highly required because it can reduce this time wastage and economical wastage and even can assist special set of people in the domestic society. And also providing this kind of services people can earn lot of money even as a profession.

In this situation particular service requesters and providers may be located in different locations and their positions may be always changing dynamically. And also there are lots of uncertain parameters such as not awareness of the required service, disturbance of the environments, the quality of the service providers' services and urgency of the service, which should be considered. Handling this kind of problems in distributed , dynamic and complex environment with involve uncertainty is a very crucial process [4]. Problem solving by message passing is the main concept of Multi-agent systems and their intelligence is an emergent feature and has strong connectivity with the agents in the system [5]. Multi-agent systems (MAS) are coming from the distributed artificial intelligence and it gives an alternative way of designing distributed control systems with autonomous and cooperative agents. Multi agent technology is the perfect method for solving this kind of complex problems [6], [7]. We have done a research to solve above described distributed service problems using multi-agent system technology. There mobile app and multi agent based solution engine have been developed for users to request and have services.

1.2 Aim and Objectives

Aim of our research to implement an intelligent system using multi agent technology to serve people in an emergency service break down in fields such as water, electricity, fire and gas in a domestic environment to find a service person for a technically non competent person or a disable person or even for technically competent person who cannot take quick decisions. The objectives of our research are as listed bellow

1. Critical review of distributed services system challenges.
2. Indepeth study of technology used for distributed service management.
3. Design and develop multi agent – based distributed service solution.
4. Evaluate the solution.
5. Apply the solution in real world.

1.3 Background and Motivation

The main challenge in distributed service system is how to direct people to their destination by reducing the transportation time and ensuring efficient use of available resources properly. The traffic condition on the road always changed and it is dynamic and complex to count when designing this kind of service systems. Because of that reason in the last few years, the number of papers devoted to applications of agent-based technologies to traffic and transportation engineering has grown enormously [8]. Therefore lots of studies have been carried out in the world about how to build intelligent transportation system. Indeed, in these days a wide spread of the Global Position System (GPS) and communication technologies (e.g. GSM, GPRS and UMTS network) has led to the implementation of interesting Intelligent Transportation System (ITS) [9] as well as distributed service systems.

Finding optimal path on the network is very crucial process because there we have to consider lots of parameters like traffic conditions, distance and heuristics like rating. In that scenario Artificial intelligent methods like A* algorithm can be used with Multi agent ethnology [10], [11].In this problem domain shortest path concept must also be considered and for that most of researchers have used famous algorithm Dijkstra Algorithm for the purpose of finding shortest path [12]. PlaSMA multi-agent simulation platform, have been used in the world to implement

Multi agent-based cost intensive Parallel Shortest-path Search [13].The benefit of this method is it has used Multi agent technology to give cost effectiveness.

There has been carried out a lot of research about similar systems like taxi dispatch system in the world and Multi agent technology has been used for those researches wildly [2].Mobile devises based distributed services have been popular and researches are currently engaging in that field [14], [15]. Most of the researches are limited to taxi dispatch systems and it is hard to find a system which kind of an assisting system for special set of people in domestic environment such as technically non competent people and technically competent people who may be disable or cannot take quick decisions.

Therefore using Multi agent ethnology in the domain of AI technics, we can solve complex problem such as distributed services problem and also can assist people in domestic environment.

1.4 Problem in brief

In domestic environment most of the people are un-aware about domestic systems or technologies and also there are disable people who know the technology but unable to resolve a problem in an emergency situation. Sometime even technically competent person cannot take quick decisions in such emergency situations. Problem is how to implement a method to meet those people who are requesting service and providing service time and cost effectively

1.5 Proposed solution

According the study of literature we found, there are distributed service systems in the word but they are mainly limited to some specific domain such as taxi hires and hard to find a service system to serve in a service break downs in domestic systems. Therefore here we propose Multi-Agent based solution to solve this described distributed service problem. Software (mobile app) is developed for user to request particular domestic service and system will find the best service and best person who should be most suitable to serve and the request is directed to him. Those information is displayed on both requester's and service provider's mobile phone. These service providers must be initially registered in the system and may be distributed over the island and also they are waiting for a request from a client.

1.6 Resource requirements

Server side:

Computer Hardware Requirements

- Laptop or PC with i5 processor
- Minimum 8GB RAM

Software Requirements

- Windows 7 or above OS
- Jboss 6.2 or Tomcat Application server
- JDK 1.8
- Spring 3.0 or above
- Spring boot Eclips, Jboss Developer studio or NetBeans IDE
- Android studio latest version
- Database Servers (MySQL)
- Jade or similar framework
- JPA 2.0 or latest version
- Microsoft Office (for documentations).

Client side:

Hardware Requirements

- Smart phone with minimum Processor. 1.4GHz, RAM. 1GB
- Storage. 16GB,
- OS. Android version 5.0 or newer

1.7 Structure of the thesis

This thesis has been structured with 8 chapters. Chpater 1 gave an overall introduction of the project. Chapter 2 provides a critical review of the developemts and issues in the area of distributed service systems by defining the research problem and identification of technologies. Chapter 3 is on technology adapted to building the multi agent solution for distributed services systems. Chapter 4 presents our approach to multi-agent based solution for distributed service systems. Chapter 5 is about the design of the proposed solution and it describes the modules and the connections. Chapter 6 presents the implementation details of each module and connections in the design. Chapter 7 will describe how the system works in real world with some screen shots and Chapter 8 gives the evaluation details such as experimental design, test cases, and evaluation strategy and data collection. Finally Chapter 9 concludes the thesis with a note on the possible further work. In Reference section it lists information of lots of referred research papers and articles and also this thesis includes an appendix section and there all supplementary information are mentioned.

1.8 Summary

This chapter provided an introduction to the entire project. For this purpose, we have presented our research problem, aim and objectives, technology used, proposed solution and resource requirements. Next chapter provides a detailed critical review of distributed service systems. And it will extract technology and define particular research problem finally in that section.

Chapter 2

Development and challenges in Distributed Services Systems

2.1 Introduction

Chapter 1 gave an introduction to overall project. This chapter presents a critical review of literature on distributed services systems. Here we formatlate our research problem and highlight the technology adopated towards a solution. In doing so, this chapter has been strctured with three sections, namely, gestation of distributed service systems, major developments in this domain, future trends of distributed service systems and problem definition.

2.2 Gestation of Distributed Service Systems

There is high demand for the optimized services today in the world. Because most of people are busy with their day today activities and they need to save their valuable limited time available. And also people wants to save their money. Therefore lots of researches involved doing research especially in transportation area to optimize the traffic and transportation system. Because for most of the services traffic and transportation domain is directly affected and optimality is required. AI technics such as Multi agent Technology is used in most of this kind of researches.

Early days Bo Chen studied about the Applications of Agent Technology in Traffic and Transportation Systems and in his research he describes about Freeway Traffic Management [8]. His system is composed of two interacting real time decision support agents, that is a freeway agent and an arterial agent, for collecting information and analysis of congestion and for generation of suitable responses. He has also studied about agent-based systems for air-traffic control and management and for railway transportation. Karsten Hager studied and has proposed some simulation in transportation domain using multi agent based solutions [16]. Purpose of his model focuses on the alteration of the domestic mobility system when newly built residential or industrial areas or rehabilitated areas are connected to the main transport system. He expects it to be a beneficial tool for researchers and urban designers to forecast mobility demands for city areas. His goal is to set up an agent based model using MATSim for the city of Stuttgart.

Jeffrey L. Adler described about A multi-agent approach to cooperative traffic management and route guidance [17]. In his paper he explained the use of cooperative, distributed multi agent systems to enhance dynamic routing and traffic congestion management. On the supply side, real time control over the transportation networks achieved through a multi agent based distributed order of system worker nodes. Distribution of traffic advisories and allocation of network capacity are performed by agents that act on behalf of information service persons. Preferences and needs of drivers are represented by agents included in intelligent in vehicle route guidance systems. Negotiation between internet service provider and driver agents looks a more efficient route allocation across space and time. Results from this simulation tests suggest that negotiation can increased driver satisfaction and achieve more optimal network performance.

Lei Zhang describe a model with a topic "Determinants of Route Choice and the Value of Traveler Information" [18]. In his model two sets of statistical models are derived. The first model explained how route preferences of drivers vary with the availability and correctness of information, while controlling and managing for observed or allocated route attributes such as travel time, number of stops, stopped delay, specific route, car (model, make, age of the vehicle etc.), and demographics (male/female ,age, household size etc.). The second model is correlates drivers' propensity to the usage of traveler information with the quality of information and drivers' attitudes, socioeconomic, demographic, travel behavioral, and other factors.

Rothkrantz has done a research about dynamic routing using the network of car drivers [19]. The goal of his project "SmartRoad" was to design ICT based solutions for traffic control as well as warning systems. In this paper he mainly focuses on dynamic routing systems. The basic idea is that car drivers communicate with each other via an intelligent system implemented on the network of the city. By tracking and communicating individual car drivers this system gets real time information of traffic speed on the roads. Based on that information he has designed dynamic routing algorithm, which is based on the concepts of artificial life. The Ant Based Control (ABC) routing algorithm process and computes the shortest traveling time from start to destination node. A simulation environment has been developed and experimental results of static and dynamic routing is presented in his research.

Bo Chen describes how to integrate mobile agent technology with multi-agent systems (MAS) to enhance the ability of the traffic congestion management systems to deal with the complexity and uncertainty in a dynamic environment [20].In his project he has developed an IEEE FIPA compliant mobile agent system named Mobile-C and designed an multi agent based real time traffic detection and management system (ABRTTDMS). This system get the advantage of the features of both stationary agents and mobile agent's technologies. The use of mobile agents allows this system to dynamically deploy new control and detection algorithms and operations to respond un-seen events and conditions. And also mobility reduces incident response time and data transmission over the connected network. The simulation of using mobile agents for dynamic algorithm and operation deployment exhibits that mobile agent approach gives great flexibility in managing dynamics in complex systems with involve lots of uncertainty.

The main algorithmic steps in this model are,

- Collect real time traffic data and process.
- Analyze and identify re identification of vehicle, estimate travel time as well as density on a freeway segment and detect incidents.
- Dynamically select suitable algorithm which match to particular incident.
- Track and monitor real time traffic conditions and notify transportation management center.

Farnaz Derakhshan designed and implemented an agent-oriented system to prioritize public transport, particularly buses in urban intersections [21]. His model plays an important role giving priority to buses in saving bus services against the effects of traffic jam, improving the frequency, speed and reliability.

When we consider about these early researches, we can see there had been high demand for optimized services especially in transportation domain. And also they have used multi-agent technology for building these optimized distributed service systems.

2.3 State of the Art of Distributed Service Systems

Today Singapore like countries practically use taxi dispatch systems which has been developed doing lots of research how to give optimized service to people and save their valuable time. The main problem is, taxis are not distributed optimally within the city and are unable to find unserved passengers effectively. Away of improving taxi operations is to deploy a taxi dispatch system that matches the vacant taxis and waiting passengers while considering the search friction dynamics. Mohen Ramezani did research and introduce method network-scale taxi dispatch model that takes into account the interrelated impact of normal traffic flows and taxi dynamics while optimizing for an effective dispatching system. The proposed model builds on the concept of the macroscopic fundamental diagram (MFD) to represent the dynamic evolution of traffic conditions. The model considers multiple taxi service firms operating in a heterogeneously congested city, where the city is assumed to be divided into multiple regions each represented with a well-defined macroscopic fundamental diagram [1].
At the same time Der-Horng Lee Department of Civil & Environmental Engineering National University of Singapore researched and found method as Advance Booking Chain Dispatching Strategy (ABC-DS).There customer can book taxis through mobile devices such as phones. Two types of bookings are commonly known: Current Booking (CBK) is one and there, the customer makes a booking call for a taxi that can reach him/her as early as possible. The other one is the Advance Booking (ABK), the customer makes a booking call and indicates the pickup time which is normally in half an hour or later. Here taking the taxi by making either CBK or ABK is defined as the 6 Booking Taxi Service (BTS) while taking the taxi by either waiting at taxi stand or hailing on seven the street is defined as the Non-Booking Taxi Service (NBTS) [3].
Abbas M. Al-Bakry from University of Babylon College of Science Technology has done a research on “Finding the Best Path Routing Using Multi-agent System” and in his research he presents a new approach to accommodate routing demands in a highly dynamic network where segment status is rapidly changing to find the best path, the router does not need to look at the network as a whole, as it is the case in routing algorithms designed so far. The proposed system consists of several agents (multi-agent) shared by some types of information for the

purpose of clarifying the case of the network in general in the way to suggest the best path, which is used by the message from the source to the destination. The proposed system has been implemented using java language and JADE platform [22].

Laxmana Siridhara Arigela has done a research on “Mobile Phone Tracking & Positioning Techniques” and this gives new idea and new trend to multi-agent based researches. Because in node or objects positioning domain mobile device and tracking system is most essential. In his research he describes about Mobile positioning technology. Mobile positioning in cellular networks will provide several services such as, locating stolen mobiles, emergency calls, different billing tariffs depending on where the call is originated, and methods to predict the user movement inside a region. The evolution to location-dependent services and applications in wireless systems continues to require the development of more accurate and reliable mobile positioning technologies. The major challenge to accurate location estimation is in creating techniques that yield acceptable performance when the direct path from the transmitter to the receiver is intermittently blocked [23].

Shih-Fen CHENG from Singapore Management University has done a research on ‘A Service Choice Model for Optimizing Taxi Service Delivery’ and his main aim was to how to optimize services to have good customer experiences. In Singapore taxi service is very important thing and its efficiency is essential. Once doing his analysis he found that there is inefficiency in organizing taxis. In most cities, taxi services are delivered either by pre-arranged pick-ups (e.g., Dial-a-Cab service in UK and Singapore), street pick-ups, or taxi stand pick-ups. This paper describe how to optimize the taxi service using latest technology like GPS. Taxi service has undergone radical revamp in recent years. In particular, significant investments in communication system and GPS devices have improved quality of taxi services through better dispatches [24].

MULTI-AGENT SYSTEMS: TRAFFIC CONTROL APPLICATION has been developed by COBEANU from Bulletin of the Transilvania University of Braşov, and there Multi-agent system technology has been proposed to model any systems who's resources are distributed (systems that can be found everywhere in the real world). Through system modelling using this technology the system's flexibility and capacity to resolve unpredicted situations that appear inside it, have been improved. In this article multi-agent systems technology advantages and applications are presented. He suggest and explain Multi-agent systems can model systems from various domains such as transportation industry, healthcare, military [25]. J.M. Corchado and D.I. Tapia and J. Bajo who are from Departamento Informática y Automática, University of Salamanca, presents a research on ‘A Multi-Agent Architecture for Distributed Services and Applications’ and this paper describe multi agent system usage in distributed service environment. Ambient Intelligence has acquired great importance in recent years and requires the development of new innovative solutions. This paper presents a novel architecture which facilitates the integration of multi-agent systems, distributed services and applications to optimize the construction of Ambient Intelligence environments. The architecture proposes a new and easier method to develop distributed intelligent ubiquitous systems, where applications and services can communicate in a distributed way with intelligent agents, even from mobile devices, independent of time and location restrictions [26].

Salima Hassas from Universit´e de Lyon, F-69000, Lyon, France has done a research on ‘Multi-Agent Dynamic Coupling for Cooperative Vehicles Modelling’ and there Cooperative Intelligent Transportation Systems (C-ITS) are complex systems well-suited to a multi-agent modelling. This research paper propose a multi-agent based modelling of a C-ITS, that couples 3 dynamics (physical, informational and control dynamics) in order to ensure a smooth cooperation between non cooperative and cooperative vehicles.it communicates with each other (V2V communication) and the infrastructure (I2V and V2I communication). This presents multi-agent model, tested through simulations using real traffic data and integrated into extension of the Multi-model Open-source Vehiculartraffic SIMulator (MovSim) [27].

Bratislav Predic, Dejan Rancic and Aleksandar Milosavljevic have done a research on the topic of ‘Impacts of Applying Automated Vehicle Location Systems to Public Bus Transport Management’ and they describe in their paper the proliferation of cheap and compact Global Positioning System (GPS) receivers has led to most Automatic Vehicle Location (AVL) systems today relying almost exclusively on satellite-based locating systems, as GPS is the most stable implementation of these. This paper presents the characteristics of a proposed system for tracking and analysing public bus transport in a typical medium-sized city and contrasts the characteristics of such a system to those of general purpose AVL systems [28].

‘TraSMAPI: An API Oriented Towards Multi-Agent Systems Real-Time Interaction with Multiple Traffic Simulators’ research did by Ivo J.P.M. Timóteo and Miguel R. Araújo from Artificial Intelligence and Computer Science Lab Faculty of Engineering, University of Porto for simulating traffic management system and this paper describes about TraSMAPI (Traffic Simulation Manager Application Programming Interface) which is designed to provide real-time interaction with Traffic Simulators using Multi-Agent Systems. It is presented as a tool for the simulation of dynamic control systems in road networks with special focus on Multi-Agent Systems. The abstraction over the simulator opens up the possibility of running different traffic simulators using the same API allowing the comparison of results of the same application in different simulators. The proposed approach is, therefore, expected to be a key asset in supporting and enhancing engineers and practitioners to make more effective control decisions and implement more efficient management policies while analyzing and addressing traffic related problems in urban areas [29].

D. Grendár, V. Wieser, J. Dúha, M. Bahleda and R. Odrobiňák have done a research and he describes the application of Positioning Systems in Intelligent Transport Systems. In this paper discuss Intelligent Transport System (ITS) and role of positioning systems in ITS. At the present there exist two different positioning systems. The first ones are satellite-based systems and the second ones are terrestrial-based systems. Each system has a different properties. The properties in terms of cost, accuracy and coverage for each of these two systems will be discussed. Then it will be described the main types of these systems and the positioning techniques in mobile networks. In conclusion UMTS and GPS/GNSS will be compared and the benefits of use of UMTS in conjunction with GNSS will be mentioned [30].

Application of multi agent system technologies in real life domain is explained in research ‘Multiagent Optimization System for Solving the Traveling Salesman Problem (TSP)’ of Xiao-Feng Xie. This paper describes multi agent optimization system (MAOS) is a nature-inspired method, which supports cooperative search by the self-organization of a group of compact agents situated in an environment with certain sharing public knowledge. Moreover, each agent in MAOS is an autonomous entity with personal declarative memory and behavioral components. In this paper, MAOS is refined for solving the traveling salesman problem (TSP), which is a classic hard computational problem. Based on a simplified MAOS

version, in which each agent manipulates on extremely limited declarative knowledge, some simple and efficient components for solving TSP, including two improving heuristics based on a generalized edge assembly recombination, are implemented [31].
Here we described about the current trend and practically used distributed service systems which are especially in taxi dispatch domain and they have used multi- agent technology as technology mainly in their systems.

2.4 Future trends in Distributed Service Systems

Using technologies like Global Positioning Systems (GPS) which is used to find the current location of a person, are used to give best optimized services for people in the society. These technologies are being used mainly in Automatic Vehicle Location and Dispatch Systems (AVLDS) these [32]. Enabling the positioning and tracking of mobile phones has emerged as a key facility of existing and future generation mobile communication systems. This feature provides opportunities for many value added location-based distributed services and systems. For instance, mobile phones are increasingly employed in traffic information systems and also present several benefits over traditional sensor-based traffic systems [14].
Lots of researches are now carried out about mobile phone positioning systems. Because for the distributed services requirement the future trend is using mobile devices and build systems.The convergence of mobile and technologies has now become the major driving force behind the standardization of third generation (3G) communication systems. The next generation mobile system is known as the Universal Mobile Telecommunications System (UMTS). UMTS is building on the success of the second generation mobile network GSM and its derivatives, HSCSD, GPRS and EDGE, to provide circuit and packet switched services to the Mobile User [33].

Here this table describes the overall summary of our literature review.

#	**Summary**	**Remark**
01	Bo Chen studied about the Applications of Agent Technology in Traffic and Transportation Systems and in his research he describes about Freeway Traffic Management.	Multi- Agent technology has been used but limited to traffic and transportation domain
02	Jeffrey L. Adler described about A multi-agent approach to cooperative traffic management and route guidance. He explained the use of cooperative, distributed multi agent systems to enhance dynamic routing and traffic congestion management. On the supply side, real time control over the transportation networks achieved through a multi agent based distributed order of system worker nodes.	Multi- Agent technology has been used and dynamic routing has been enhanced but limited to traffic and transportation domain
02	Rothkrantz has done a research about dynamic routing using the network of car drivers. The goal of his project "SmartRoad" was to design ICT based solutions for traffic control as well as warning systems.	This is a traffic control system and Multi- Agent technology has been used
03	Bo Chen describes how to integrate mobile agent technology with multi-agent systems (MAS) to enhance the ability of the traffic congestion management systems to deal with the complexity and uncertainty in a dynamic environment.	Mobile agent technology has been integrated with Multi- Agent technology.
04	Farnaz Derakhshan designed and implemented an agent-oriented system to prioritize public transport, particularly buses in urban intersections.	Traffic control system and it has used Multi- Agent technology.

05	D. Grendár, V. Wieser, J. Dúha, M. Bahleda and R. Odrobiňák have done a research and he describes the application of Positioning Systems in Intelligent Transport Systems.	Vehicle positioning system and it has used Multi- Agent technology.
06	Abbas M. Al-Bakry from University of Babylon College of Science Technology has done a research on “Finding the Best Path Routing Using Multi-agent System” and in his research he presents a new approach to accommodate routing demands in a highly dynamic network	Best path finding method with multi agent technology on a network of nodes.
07	Shih-Fen CHENG from Singapore Management University has done a research on ‘A Service Choice Model for Optimizing Taxi Service Delivery’ and his main aim was to how to optimize services to have good customer experiences	Optimizing taxi service using multi-agent technology.
08	MULTI-AGENT SYSTEMS: TRAFFIC CONTROL APPLICATION has been developed by COBEANU from Bulletin of the Transilvania University of Braşov, and there Multi-agent system technology has been proposed to model any systems who's resources are distributed (systems that can be found everywhere in the real world).	Traffic control system with multi-agent technology.
09	Salima Hassas from Universit´e de Lyon, F-69000, Lyon, France has done a research on ‘Multi-Agent Dynamic Coupling for Cooperative Vehicles Modelling’ and there Cooperative Intelligent Transportation Systems (C-ITS) are complex systems well-suited to a multi-agent modelling	Multi-agent model for dynamic routing

10	Bratislav Predic, Dejan Rancic and Aleksandar Milosavljevic have done a research on the topic of ‘Impacts of Applying Automated Vehicle Location Systems to Public Bus Transport Management’	Application of automated vehicle location system using multi-agent technology.
11	Laxmana Siridhara Arigela has done a research on “Mobile Phone Tracking & Positioning Techniques” and this gives new idea and new trend to multi-agent based researches	Mobile Phone Tracking & Positioning Techniques with multi agent technology

Table 2.4.1. *Summary information of literature review*

2.5 Problem definition

According to the critical literature review we found, though there are lots of researches have been carried out to give distributed services, they are basically limited to vehicle hire, traffic management and transportation fields. And also there are lesser number of systems in this domestic domain, which supports technically non competent or disable people in the society. In this research main objective is to build distributed service system using multi agent technology that can serves domains such as vehicle / three wheel hiring, technician’s service from carpenter/plumber, goods transport, ambulance service, tuition classes and consultancy etc.

2.6 Summary

This chapter provided detail description about the literature review, how previous researches addressed this distributed services problem and what type of systems they have developed and what type of technology they have used. Some limitations and drawbacks of available systems were also described in that section. And also it described current usages and the future trends of the distributed services system field. Next chapter provides a detailed explanation of the technology section of our research project.

Chapter 3

Multi Agent Technology

3.1 Introduction

Chapter 2 formulated our research problem and highlights the technology adopted towards a solution. In this research mainly identified technologies are Multi Agent System technology, Global Positioning System (GPS) technology and Google Map APIs. Using GPS technology requester's and service providers' current location is captured and Google Map Platform APIs is used for creating the network of nodes of requesters and service providers. Multi Agent technology is used for solving the network and finding the best service provider to serve the requester. In addition to these technologies MySQL Database technology, Android architecture and components, Spring boot and Service architecture is also explained in this section.

3.2 Multi Agent System Technology

Handling distributed service management kind of problems in dynamic, distributed and complex environment with involve uncertainty is a very crucial process [4]. Problem solving by message passing is the main concept of Multi-agent systems and their intelligence is an emergent feature and has strong connectivity with the nodes/agents in the system [5]. Multi-agent systems (MAS) are derived from the distributed artificial intelligence and it allows an alternative way to design distributed control systems based on autonomous and cooperative agents, exhibiting modularity, robustness, flexibility and adaptability. Multi agent technology is the perfect method for solving this kind of complex problems because features described [6], [7].

Multi-agent systems can be used to solve problems that are difficult or impossible for an individual or a single agent or a monolithic system to solve. There are some specific features in the agents in Multi Agent system. Below are some of key characteristics of intelligent agents in multi agent systems [34].

1-Autonomy. Agent can exhibit control, act independently in his internal state.

2- Reactivity: Agent maintains an ongoing interaction with its environment, and sense to changes that occurs in the environment**.**

3- Pro-activeness: Agent is capable of achieving own defined goal.

4- Social Ability: Agents has ability to interact with other via agent-communication language, and cooperate with other agents in the environment.

5- Capacity for Cooperation: intelligent agents co-operates with other agents in the environment achieving of own objectives.

6- Capacity for Reasoning: Intelligent agent can express the ability to infer and extrapolate with available current knowledge and experiences.

7- Adaptive Behavior: Based on previous experience, agent learns their behaviors.

8- Trustworthiness: Client must be highly confident that its agents will act and report truthfully.

The most important factor in multi agent system is knowledge sharing and exchange. An agent is usually described as a persistent entity with some degree of independence or autonomy that carries out some set of operations depending on what he perceives. And also an agent usually contains some level of intelligence. Therefore he has to have some knowledge about its goals and desires/targets. The whole multi-agent system is created to be capable of reaching goals that are difficult to achieve by an individual agent or a monolithic system. In multi- agent systems, an agent usually cooperates with other agents, so it should have some social and communicative abilities.

In next 20 years in future, computers will become so cheap that they will be everywhere and ultimately every product will have use at least one computer chip. More complex items like vehicles will have hundreds of chips of computers, if not thousands or more, computer chips each controlling a different aspect of the vehicle or machine. Computers will become part of our life, such as we need oxygen for breathing. Then how will these computers interact and communicate with each other? How will they work together, solving common goals, to fulfill human needs and desires? A distributed system using Contract Net Interaction Protocol can solve these problems in computer world. A distributed system could literally connect every person to every other entity creating an intelligent global net. The possibilities of this distributed network compared to the current internet would be like comparing the modern computer to the abacus used in ancient times [35]. Contract Net specifies the interaction between agents for fully automated competitive negotiation through the use of contracts.

3.3 JADE- Java Agent DEvelopment Framework

JADE is open source platform developed for peer to peer agent base software application. JADE is implemented using Java language and gives library files for java developers to use. It support and make easy of the implementation of multi-agent systems through a middle-ware that goes with the FIPA specifications and also through a set of graphical tools which support debugging in deployment stage. A JADE-based software system can be distributed across machines/platforms and the configuration can be done even in remote GUI. JADE is fully implemented in Java language and java version 5 is the minimal system requirement.

3.4 Global Positioning System (GPS) Technology

GPS system was initially developed by the US government for the purpose of military operations. Nowadays anyone with a GPS device such as mobile phone or handheld GPS unit, can access the radio signals of satellites broadcast. GPS is from a network of about 30 satellites which are orbiting the Earth at an altitude of 20,000 km. There at least four GPS satellites are visible at given time for any one on the earth. Each of them transmit information about the current time and the position. These signals are travelling faster at the speed of light and they are intercepted by our GPS device, and then it calculates how far away each mentioned satellite is based on how long it took for the messages to come. With these information and calculations our GPS device can point our location.

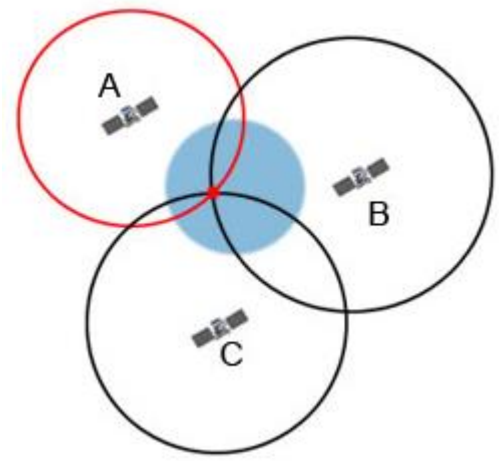


Figure 3.1 Satellites View for GPS Technology

3.5 Google Map Platform

Google map platforms provides lots of APIs to get various information on the Google map like traffic conditions, directions (Provide directions for transit, biking, driving, or walking between multiple locations), distance matrix (Calculate travel times and distances for multiple destinations) and roads (Determine the precise route a vehicle travels).In this research we use distance matrix API for calculate travel times and distances for multiple destinations.

Distance Matrix API:

The Distance Matrix API is a service that provides travel distance and time for a matrix of origins and destinations. To this service, as input parameters we pass latitude/longitude coordinates of origin and one or more destination nodes with the travel mode (driving, walking, bicycling or transit). Then the service will provide travel distance and time values in a JSON or XML form.

3.6 Android Architecture and Components

The components in Android Architecture are a collection of libraries that support us design robust, testable, and maintainable mobile apps with more power over lifecycle management and data persistence. Android use Model-View-ViewModel architecture and here it describes main component in this architecture.

1. Model is the data layer of our app. It abstracts the data source.
2. View contains the UI of our mobile app. Most often it’s implemented as an Activity or Fragment. View informs ViewModel of user interactions and displays results received from the ViewModel. View should be lightweight and contain very little business logic.
3. ViewModel serves as a bridge between your View and Model. It works with the Model to get and save the data. The View observes and reacts to the data changes exposed by the ViewModel.

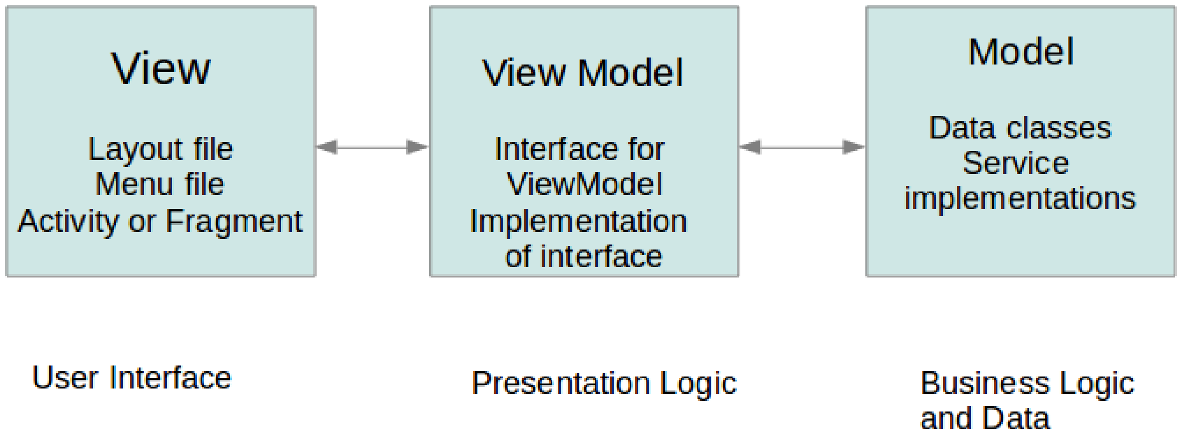


Figure 3.6.1 Android Model-View-ViewModel Architecture

- Activity/Fragment − Controls all aspects of the application lifecycle and activity stack.
- View - The view is responsible for handling components such as Menus, Permissions, Event listeners ,Showing dialogs, Working with Android View and Widget, Starting activities, All functionality which is related to the Android Context

- View Model - The view model contains the data required for the view. It is an abstraction of the view and exposes public properties and commands. It uses observable data to notify the view about changes The view model do these activities Exposing data ,Exposing state (progress, offline etc.), Handling visibility, Input validation, Executing calls to the model and Executing methods in the view.
- The Mode - It contains a data provider and the code to fetch and update the data. The data can be retrieved from different sources, such as REST API. Realm db, SQLite db, Handles broadcast etc.

3.7 MySQL Database Technology

Database is used to store collection of data and it is a separate application .Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds (Create, Insert, Update, Delete etc.). Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those type of systems. These days, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as Foreign Keys. A Relational Database Management System is a software that has features to,

- Implement a database with tables, columns and indexes
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

MySQL is an open source relational database management system based on Structured Query Language (SQL). MySQL runs on virtually all platforms, including Linux, UNIX and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web applications and online publishing**.**

3.8 Spring Boot and Service Architecture

3.8.1 SOA - Service Oriented Architecture

A Service Oriented Architecture is a software architecture pattern, which application components provide services to other components via a communications protocol over a network. The communication can involve either simple data passing or it could involve two or more services coordinating connecting services to each other. Services such as RESTful Web services carry out some small functions, such as validating an order, activating account, or providing shopping cart services.
There are 2 main roles in SOA, a service provider and a service consumer. A software agent may play both roles. The Consumer Layer is the point where consumers (human users, other services or third party application). Interact with the SOA and service provider layer consists of all the services defined within the SOA.

3.8.2 Microservices Architecture

Microservices must be a real need in the system architecture as it could be designed wrongly. It means a service should be independently deployable, or be able to shut-down a service when is not required in the system and that should not have any impact on other services. The main idea in this architecture is to business logic is break down to small units and implement. And those service may be acting independently.

3.8.3 Spring Boot

Spring Boot is a project built on the top of the Spring framework. It provides a simpler and faster way to set up, configure, and run both simple and web-based applications. In the Spring core framework, we need to configure all the things for our self. Then we can have a lot of configuration files, such as XML descriptors. That’s one out of the main problems that Spring Boot solves for us. It smartly chooses our dependencies, auto-configures all the features we want to use, and we can start our application with one click. Furthermore, it also simplifies the deployment process of our application. Spring Boot enables building production-ready applications quickly and provides non-functional features such as,

- Embedded servers which are easy to deploy with the containers(Inbuilt tomcat server)
- It supports in monitoring the multiples components
- It helps in configuring the components externally

3.9 Summary

This chapter provided detail description about the technologies used in this research and the explanation for using those technologies. The main technology is multi-agent technology and other supportive technologies are JADE, GPS, and Google API, MySQL Database, Android and Spring Boot Service API technologies. Next chapter provides a detailed explanation of the novel approach of our research project.

Chapter 4

Novel Approach to Distributed Services System

4.1 Introduction

Chapter 3 explained about the Technologies used in this research project. The inspiration of this novel approach is to develop an intelligent system using multi-agent technology to assist mainly special set of people in domestic environment who are technically non competent or may be disable to solve issues in an emergency situation. Our new approach to distributed service systems is named as **MASDSS** acronyms for Multi Agent System for Distributed Services System.

In doing so, this chapter has been structured with these sections namely the hypothesis which is a tentative statement about the relationship between two or more variables of the solution, the input of the system which are the parameters to be entered by service requesters and service providers to the system, the outputs of the system which are the best solution and instructions to be displayed on the screen of the mobile phone, the process which is set of methods and actions doing towards the best solution, the overall features of the system which are the behaviors of the developed system, and users who are the actors of the system.

4.2 Hypothesis

Implement an intelligent system **MASDSS** using multi agent technology to serve people in an emergency service break down in fields such as water, electricity, fire and gas in a domestic environment to find a service person for a technically non competent person or a disable person or even for technically competent person who cannot take quick decisions.

4.3 Input of the System

These are the inputs to the system.

- Service Type: - For this input type these values can be captured, Plumber service, Carpenter, Electrician Service, Vehicle/Taxi hire , Car/Vehicle repair etc.
- Service sub type: - These are descriptive pre-defined service categories such as a Leak repair, Drain cleaning, Toilet repair, Garbage disposal repair, Water Heater problem, Sewer repair, Gas Lines repair, Kitchen sinks, Bathroom faucets & fixture, Wells and Water pumps repair, Home water supply, TV/Radio repair, Car/Vehicle repair etc.
- Current Location: - This will capture the current location attribute that is latitude/longitude values of service requester and service providers.
- Mobility of the start and destination nodes: - This indicate whether both requester and service providing person can move or only one of them can move.
- Traffic condition on the road: - That is a heuristics value to be captured from Google map.
- Rating value of the service person: - Base on the quality of the current service of particular service provider, rating is captured from the service requester and saved in database. For the next service of same service provider, model will consider that previous rating value.

4.4 Output of the System

The following items are the outputs of the system.

- Display most suitable service provider to serve the requester and notify two parties. And also on mobile screen relevant service information such as service location, unit charge for the selected service are displayed for guiding purpose.

4.5 Process

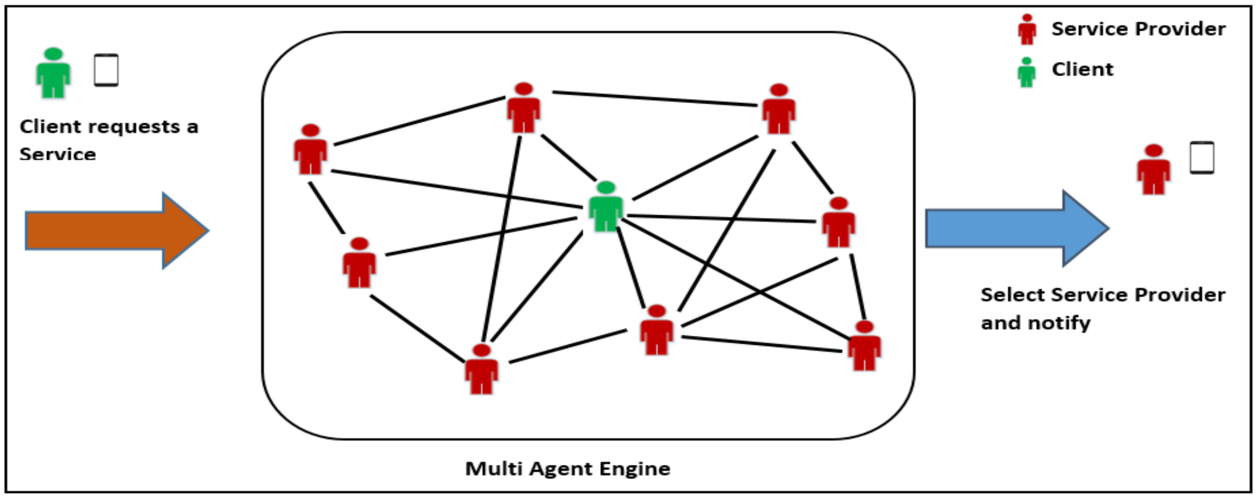


Figure 4.5.1 Top level architecture diagram

As displaying in figure 4.5.1 our propose process can be divided in to mainly three parts,

1. **Client** request particular service entering required parameters
2. **Multi agent engine** which will process input parameters and fine optimal solutions
3. Optimal solutions are notified to relevant parties as **output**

The process of the system explained the behavior of the distributed service software (mobile app).Once requester make a request on the mobile app then the request is sent to the server which may be located on the cloud. Then the algorithm gather all input parameters and initialize agents for communication.

The next step is to find the optimal solution that is the most suitable service provider to serve the requester. The system will instruct the requester mentioning the service provider information and also the place where the meeting is to be occurred. Once service is completed, client will update rating in the system with his satisfactory level for the service provider.

4.6 Overall features of the system

Below items are the main features implement in this project.

- Registered service providers can earn money giving good service: - If service providers are providing good effective and satisfactory service then client base will be increased and service providers will be able to provide more services. They will be busy and will earn more.
- Assist clients providing optimal satisfactory solution: - This system specially serves domestic people to have satisfactory service when an emergency service break down occurs.
- Accurate: - This system is accurate because it uses Multi- Agent based technologies which are proven to be accurate.
- Time saving: - People in domestic environment can save their valuable time spend for finding a service person when there is an emergency service break down, because this system assists them quickly finding best service provider.

4.7 Users of the System

Below are the users of the system.

- Registered service providers: - Service providers are to be registered in the system and kept their account active regular basis.
- Any person who needs support/services: - Any person in domestic society can use this system to find particular optimal service person for complete his request with satisfaction.

4.8 Summary

This chapter provided a detailed explanation of the approach of this project. In our novel approach we give software system for client to enter required parameters of his required service and find most suitable service and service person and same system will capture the rating value from the client for the service of the service provider. Next chapter describes the design section of our research project.

Chapter 5

Design of MASDSS

5.1 Introduction

Chapter 4 explained about the approach part of this research project. It described our novel approach with multi-agent technology solution to distributed service problem. This chapter presents the top level architecture of the project and design will explain what each process is doing in this system. Database design, Agent design such as manager agent, service requester agent service provider agent are described in this section and also the UI- user interface and API design are explained. The diagrams such as ER diagram of database design and flow chart diagram of agent design is displayed here.

5.2 Database Design

All registered service providers and their all information is recorded in MySQL database. There we have designed a database called **distributedservicesdb** and there multiple tables are designed for saving particular information in the system. ER diagram of the database design is displayed in figure 5.2.1. Here we have described some important database tables and what information is saved in those tables.

People: - All registered service providers' information is saved such as NIC, customer name, contact details etc. in this table. Once a service provider is registered in our system this table is populated with his information.

UserPool: - All current ongoing information of service providers and clients is saved in this table. Dynamic information such as agent current locations (latitude and longitude vales), traffic condition on the road, current rating values of service providers, requested service of service requesters etc.

UserPoolSub: - All current ongoing information of sub service of each service such as 'Water leak in bathroom or Pipe line damage in bathroom' of main service Plumber is saved in this table.

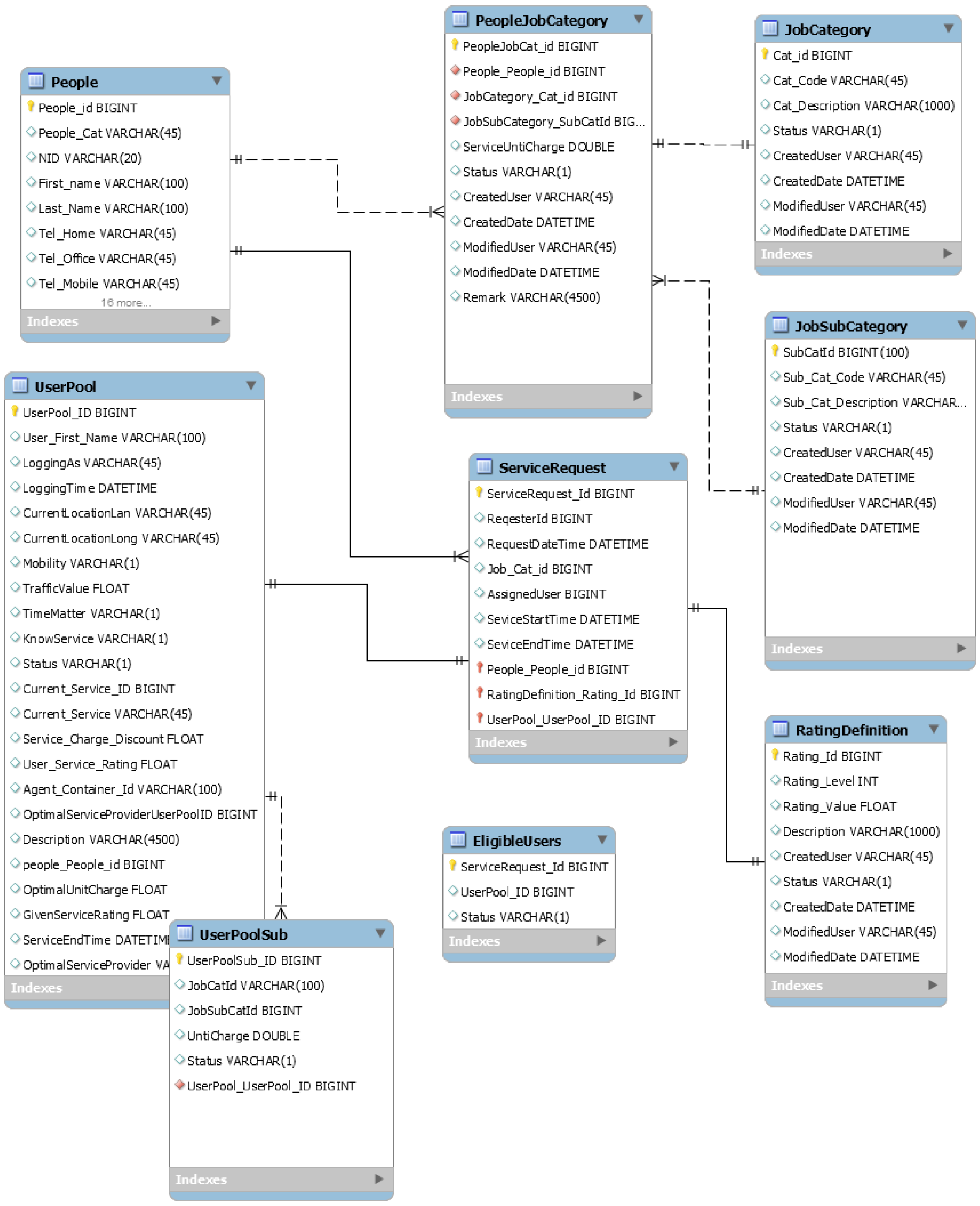


Figure 5.2.1 ER Diagram of Database Design

RatingDefinition: - Rating definitions are designed to be saved in this table.

JobCategory: - service category information such as Plumber, Carpenter, Electrician, Vehicle hire is saved in this table.

JobSubCategory: - Detailed descriptions of above job categories such as battery death problem of vehicle repair service, air condition problems of vehicle repair service, leak repair request of plumber service are designed to be saved in this table.

EligibleUsers: - Eligible service providers for particular service request is to be saved in this table.

5.3 Agent Design

This section describes how agents are designed in this model and what each agent is performing. There are three main agents in this models and they are,

- Manager Agent :- Initialize, interrupt, destroy, terminate other agents
- Service Requester Agent :- Map to real world service requesters/Clients
- Service Provider Agent :- Map to real world service persons

At first, Manager Agent will initialize other agents (service requester and service provider agents). There may be one or more service requester and service provider agents in the agent container according to the demand of requests and active service providers in the pool. After that service requester agent will broadcast CFP message to every service provider agents in agent container appending required service information to the message. Then that message will be received to each service provider agent and they will process the message body, if they are providing particular requested service they will reply with proposals (Appending the features of the service what service provider is providing). If service provider does not have requested service he will refuse the request and send inform message to requester agent. Then service requester agent will check and evaluate the proposals from service providers, if required criteria is met he will select best service provider and send order request to particular selected service provider. For this selection several communications cycles may be occurred between agents. After accepting order request, both service provider and service requester agents will be terminated by the manager agent.

5.3.1 Manager Agent

This agent can communicate and even interrupt the other agents at his wish. This does happen with other agents unless in very exceptional situations. Manger agent first query UserPool table in database and check whether there are any pending service requesters available. Then he check whether there are any service providers who are serving relevant requested service. If there are such records in UserPool table he then creates separate agent container and initialize only particular service requesters and provider agents. After finishing communications between those agents and finding optimal solution he then destroys those agents. While initialized agents are communicating each other manager agent always queries the UserPool table and check whether there are any pending service requesters available. If there exists requesters then he creates agent container and initialize new agents for newly entered service requesters. This agent acts as a filter agent also.

5.3.2 Service Requester Agent

Requester agents (Clients) are making requests with providing input parameters. Service requester agent asks service giving particular service (If client knows the required service) and detailed information such as “Plumber - need Toilet Repair service”. This agent is initialized by manager agent appending required service information. This agent starts sending message to all the service provider agents indicating the target service and waits replies from those agents. Best solution is recorded while communicating with service provider agents and optimum solution is informed to the application. Once solution is found or not found finally service requester agent will be destroyed by the manager agent.

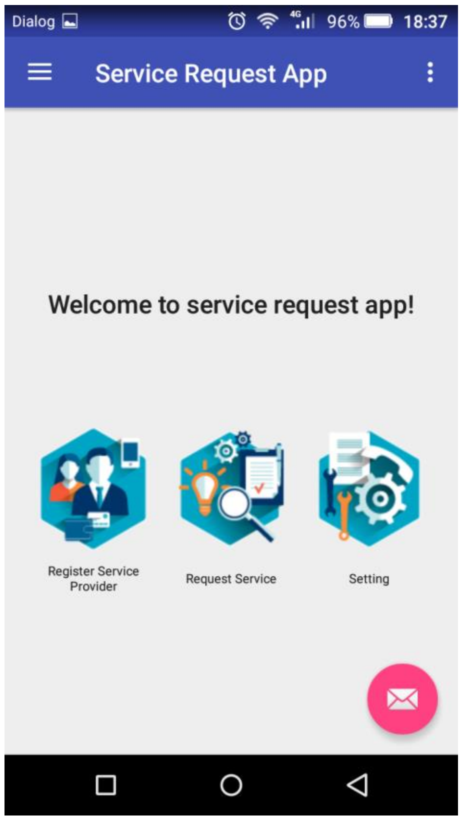
5.3.3 Service Provider Agent

Service Provider agents are distributed overs the network. They have parameter values such as current location, current status (whether attending a service or idle), providing service etc. Service Provider agents propose all providing services with all information such as service description, unit charge amount etc. This agent also will be initialized by the manager agent. This does negotiation with other agents and proposes his criteria about the providing services. Once relevant criteria is met then service provider agent will accept the request of the service requester agent and inform to the application. Once process is completed this agent also will be destroyed by the manager agent.

5.4 Application Design of MASDSS

Using android studio mobile interface is designed for clients and service providers to interact with the system. There using this interface service requester can enter particular parameter information for the required service and submit request. Once the requested is submitted agent model will choose best service provider and that information also displayed in this mobile interface. And that requested service information is displayed in service provider's screen also for getting aware about the requested service. After completing this service requester can rate service provider with his satisfactory level of given service through this provided mobile screen.

In chapter 07, it will describe about the features and operations of mobile application in-detail with sample screen layouts. In figure 5.4.1, it displays user interface design of first mobile screen.



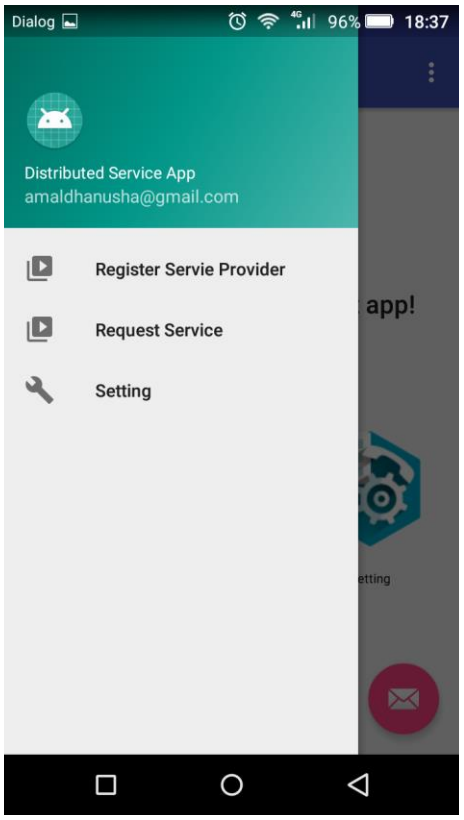
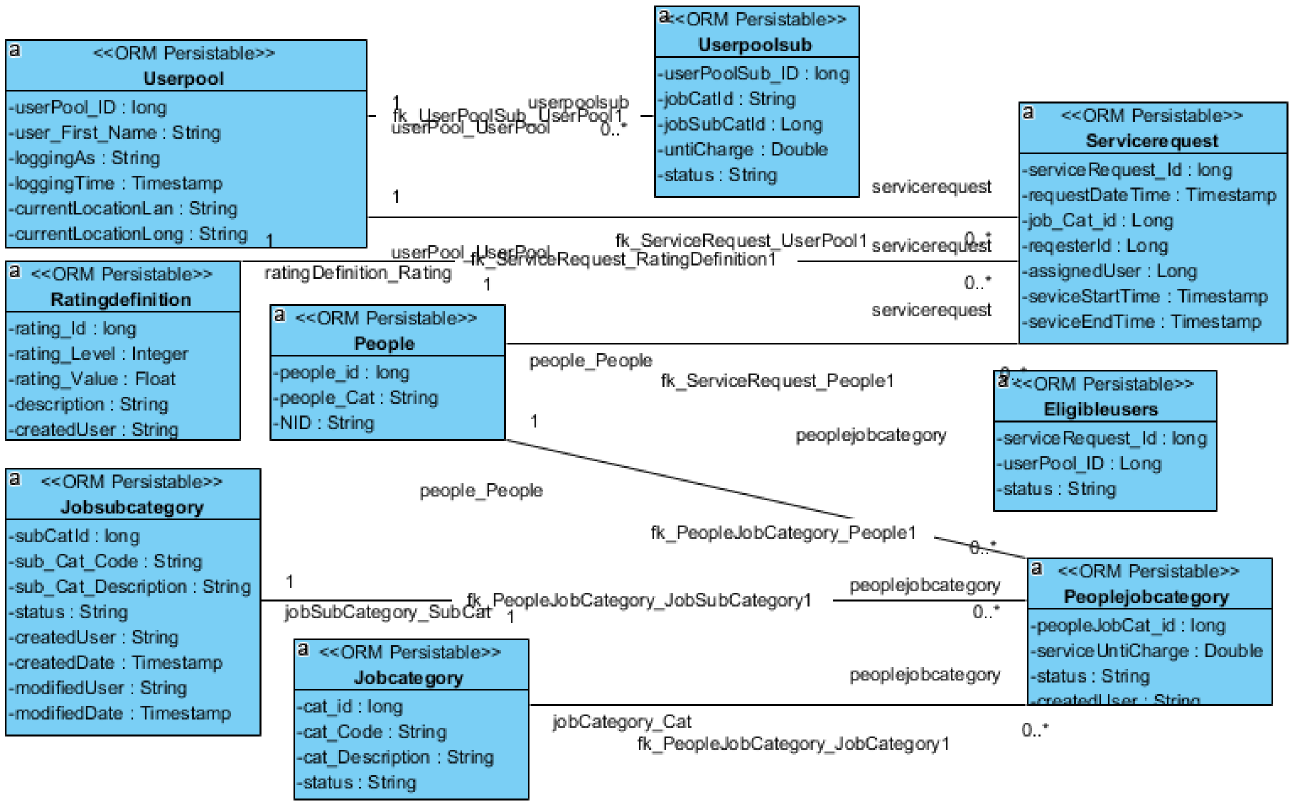


Figure 5.4.1 User Interface Design

5.5 API Design of MASDSS

Mobile interface will collect parameter information from the user and will pass those information to multi agent engine to process, and processed information will be shown in mobile UI for giving instructions to the user. Those data transferring part between mobile UI and server will be handle by set of published APIs. Those APIs are to be designed to create in spring boot application and deployed on tomcat server. The class diagram of those APIs is mentioned in below figure 5.5.1.



<<ORM Persistable>> Userpool
-userPool_ID : long
-user_First_Name : String
-loggingAs : String
-loggingTime : Timestamp
-currentLocationLan : String
-currentLocationLong : String
<<ORM Persistable>> Userpoolsub
-userPoolSub_ID : long
-jobCatId : String
-jobSubCatId : Long
-untiCharge : Double
-status : String
<<ORM Persistable>> Servicerequest
-serviceRequest_Id : long
-requestDateTime : Timestamp
-job_Cat_id : Long
-reqesterId : Long
-assignedUser : Long
-seviceStartTime : Timestamp
-seviceEndTime : Timestamp
<<ORM Persistable>> Ratingdefinition
-rating_Id : long
-rating_Level : Integer
-rating_Value : Float
-description : String
-createdUser : String
<<ORM Persistable>> People
-people_id : long
-people_Cat : String
-NID : String
<<ORM Persistable>> Eligibleusers
-serviceRequest_Id : long
-userPool_ID : Long
-status : String
<<ORM Persistable>> Jobsubcategory
-subCatId : long
-sub_Cat_Code : String
-sub_Cat_Description : String
-status : String
-createdUser : String
-createdDate : Timestamp
-modifiedUser : String
-modifiedDate : Timestamp
<<ORM Persistable>> Jobcategory
-cat_id : long
-cat_Code : String
-cat_Description : String
-status : String
<<ORM Persistable>> Peoplejobcategory
-peopleJobCat_id : long
-serviceUntiCharge : Double
-status : String
-createdUser : String
fk_UserPoolSub_UserPool1
userPool_UserPool
userpoolsub
fk_ServiceRequest_UserPool1
servicerequest
fk_ServiceRequest_RatingDefinition1
ratingDefinition_Rating
fk_ServiceRequest_People1
people_People
peoplejobcategory
fk_PeopleJobCategory_People1
fk_PeopleJobCategory_JobSubCategory1
jobSubCategory_SubCat
fk_PeopleJobCategory_JobCategory1
jobCategory_Cat
1
0..*

Figure 5.5.1 Class Diagram for API Design

5.5 Summary

This chapter provided a detailed explanation of the design of MASDSS. Chapter explained how each processes interact each other and what type of actions each process is performing. There, Database design, Agent Design, Application designs and API designs were explained. Next chapter describes the implementation of the model and application in design of this research project.

Chapter 6

Implementation

6.1 Introduction

Chapter 5 explained about the design part of this research project. That described what type of models there in design and what each model is doing. Main models of this systems are Database, Agent, User Interface and Service Model. This chapter presents the implementation of those explained models in design and it will explain how each model is performing its tasks. Database implementation, Agent implementation, UI and API implementations are organized in to sub sections of this chapter.

6.2 Setting up Development Environment

- MySQL database is created and MySQL workbench tool is installed for handing and configuring database and queries.
- XAMPP software is installed for controlling database and severs.
- Spring-Tool-Suite (STS) IDE is installed for creating the programme and source code.
- Spring Boot framework is used for developing APIs.
- Android Studio is installed for developing mobile app screens.
- Tomcat web server is installed.

6.3 Agent Implementation

We created java project as '**DistributedService'** importing jade library files. Java program is written on jade framework and it consist below main classes.

ConnectionManager: Establish database connection.

MainClass : The main class of the project

ManagerAgent : Acts as the manager agent in agent container. It extends jade agent interface. He manages other agents' life cycles.

ServiceProviderAgent : Acts as service provider agent which extends jade agent interface.

ServiceRequesterAgent : Acts as service requester agent which extends jade agent interface.
ServiceProviderGui : User interface class for registering service providers runtime.
ServiceRequesterGui : User interface class for creating service requesters runtime.
Sample source code segment for service provider agent is in appendix 03.

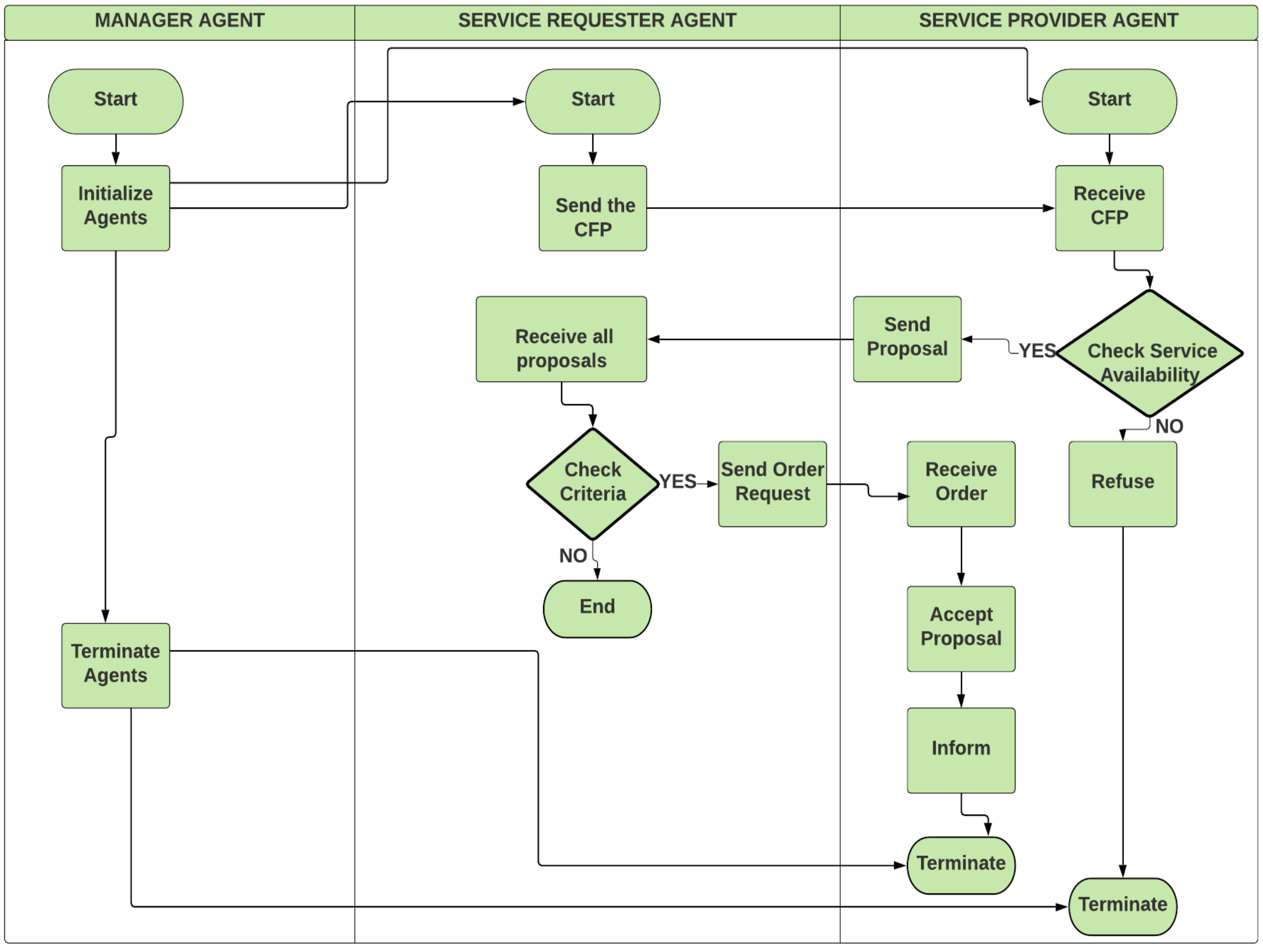


Figure 6.3.1 Flow Chart Diagram of Agent Behavior

This flow chart explains how agents are interacting each other in our multi- agent engine. There, Manager Agent will do specific tasks such as agent initialization, agent termination in the agent container.

Pseudo code segments of agent's classes are described in below subsections.

6.3.1 Agent Initialization

```
Agent set up () {
Register service provider agents in DFagent yellow page.
}
```

6.3.2 Manager Agent

```
 AGENT INITIALIZATIONS
START SETUP
   Create customer agents
   Create merchant agents   END SETUP
```

6.3.3 Service Provider agent

```
          AGENT INITIALIZATIONS
          START SETUP
          addBehaviour(new OfferRequestsServer());
          END SETUP
// This is the behaviour used by service provider agents to request
services from service requester agents
          START BEHAVIOR OfferRequestsServer
                    Receive CFP messages from customers
                    If requested services available
                       ACLMessage.PROPOSE
                    Else
                       ACLMessage.REFUSE
          END BEHAVIOR
```

6.3.4 Service Requester agent

```
AGENT INITIALIZATIONS
START SETUP
          ADD BEHAVIOR RequestService
END SETUP
START BEHAVIOR RequestService
       Send the CFP to Service Provider Agent
      Receive the list of suitable services/subservices
         Send the CFP to suitable services/subservices
         Receive all proposals/refusals from Service Provider agents
```

```
        Send the service order to the most suitable Service Provider    Receive the
service order reply
END BEHAVIOR
```

6.4 Database Implementation

In MySQL database, designed tables are created and sample script of table **UserPoolSub** is mentioned here.

```
CREATE TABLE `distributedservicedb`.`UserPoolSub` (
 `UserPoolSub_ID` BIGINT NOT NULL AUTO_INCREMENT,
 `JobCatId` VARCHAR(100) NULL,
 `JobSubCatId` BIGINT NULL,
 `UntiCharge` DOUBLE NULL,
 `Status` VARCHAR(1) NULL,
 `UserPool_UserPool_ID` BIGINT NOT NULL,
 INDEX `fk_UserPoolSub_UserPool1_idx` (`UserPool_UserPool_ID` ASC),
 PRIMARY KEY (`UserPoolSub_ID`),
 CONSTRAINT `fk_UserPoolSub_UserPool1`
  FOREIGN KEY (`UserPool_UserPool_ID`)
  REFERENCES `distributedservicedb`.`UserPool` (`UserPool_ID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
```

6.5 User Interface Implementation

Using android studio IDE mobile user interfaces are developed and sample layout and pseudo code segment is mentioned here.

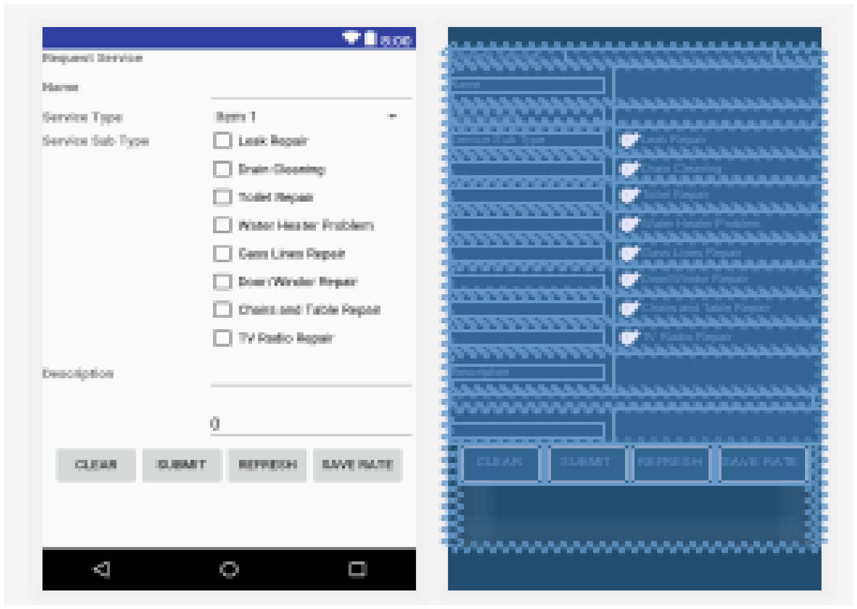


Figure 6.4.1 Sample Layout Implementation of UI

------------------------------ *Quoted from source code* ----------------------------------

```
 <LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal">
        <TableRow>
            <TextView
                android:id="@+id/tv0"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Request Service"
                android:fontFamily="Arial"
                android:textStyle="bold" />

            <TextView
                android:id="@+id/tv1"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"/>
        </TableRow>
```

---*End*---

Above source code segment will implement the layout items in Figure 6.4.1 user interface.

6.6 API Implementation

Using spring Boot IDE designed APIs are developed, deployed and published on tomcat server. This APIs are consumed by Mobile UI for interact with database and multi-agent engine. Sample pseudo code segments is mentioned here and sample URLs and json request body is mentioned in appendix 04

---------------------------- *Quoted from source code* --------------------------------

```
package com.distributedService.controller;
import java.util.Calendar;
import java.util.List;
@RestController
@RequestMapping(value="/distributedServices")
public class DistributedServiceMainController {
@Autowired
private UserPoolDao userPoolDao;
@Autowired
private JobCategoryService jobcategoryService;
@RequestMapping(value="/getJobCat" , method=RequestMethod.GET)
public ResponseEntity<List<JobCategoryResponse>> getJobCategiries(){
List<JobCategoryResponse> jobCategories = jobcategoryService.findAll();
if(jobCategories.size() > 0){
return new ResponseEntity<List<JobCategoryResponse>>(jobCategories ,  HttpStatus.OK);
}
else{
return ResponseEntity.noContent().build();
}

@RequestMapping(value="/createUserPool" , method=RequestMethod.POST)
public String createUserPool(@RequestBody UserPoolModel userPoolModel){
UserPool userPool = new UserPool();
userPool.setAgentContainerId(userPoolModel.getAgentContainerId());
userPool.setCurrentLocationLan(userPoolModel.getCurrentLocationLan());
userPool.setCurrentLocationLong(userPoolModel.getCurrentLocationLong());
userPool.setCurrentService(userPoolModel.getCurrentService());
//userPool.setCurrentServiceID(userPoolModel.getCurrentServiceID());
userPool.setKnowService(userPoolModel.getKnowService());
userPool.setLoggingAs(userPoolModel.getLoggingAs());
userPool.setLoggingTime(userPoolModel.getLoggingTime());
userPool.setMobility(userPoolModel.getMobility());
userPool.setOptimalServiceProvider(userPoolModel.getOptimalServiceProvider());
userPool.setOptimalServiceProviderUserPoolID(userPoolModel.getOptimalServiceProviderUserPoolID());
userPool.setPeoplePeopleId(userPoolModel.getPeoplePeopleId());
userPool.setTimeMatter(userPoolModel.getTimeMatter());
userPool.setTrafficValue(userPoolModel.getTrafficValue());
userPool.setServiceChargeDiscount(userPoolModel.getServiceChargeDiscount());
userPool.setDescription(userPoolModel.getDescription());
userPool.setUserFirstName(userPoolModel.getUserFirstName());
userPool.setUserPoolID(userPoolModel.getUserPoolID());
userPool.setUserPoolStatus(userPoolModel.getUserPoolStatus());
userPool.setUserServiceRating(userPoolModel.getUserServiceRating());
userPoolService.createUserPool(userPool);
return "OK";
}
```

---------------------------- *End*--------------------------------

Above source code segment implements the main rest controller of the distributed service API. It accepts request payload from mobile user interface and do below operations.

- Save particular information in main **userPool** table in the database.(POST method)
- Query saved information (GET method).
- Query best service/Service provider information (GET method).
- Update rating information (PUT method).

6.7 Summary

This chapter provided a detailed explanation of the implementation of model and application of MASDSS. It described how each model Database, Agent, UI and Service model perform designed task. In next chapter it will describe how the developed system works in live environment. It will explain with some screen shots, the features and behaviors of mobile interface and how user should operate the system.

Chapter 7

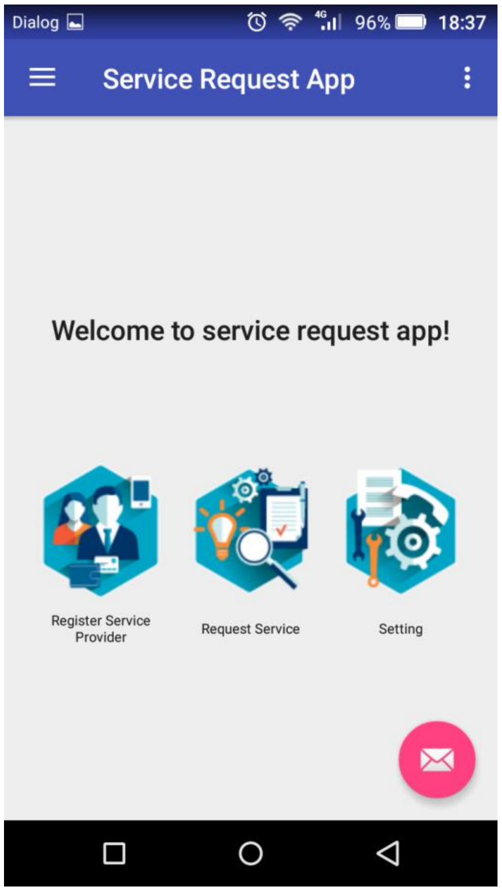
How the System Works

7.1 Introduction

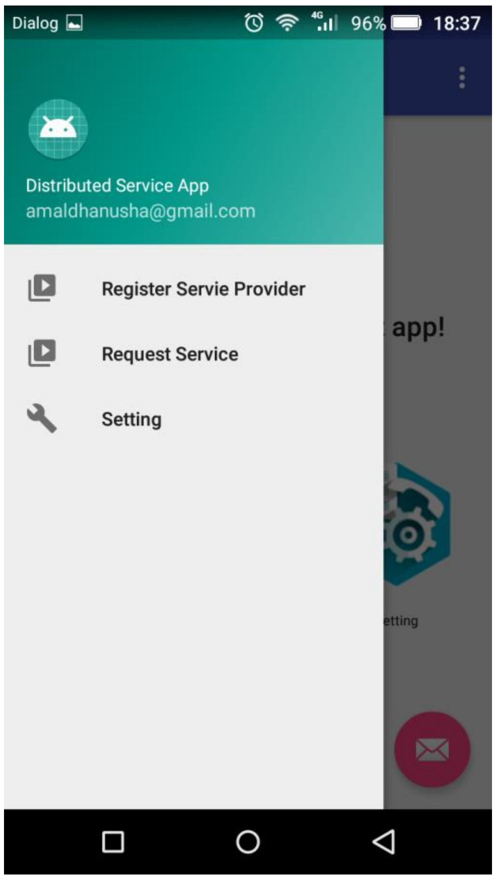
Chapter 6 explained about the detail of implementation of our research and it explained implementation of each models of Database, Agent, User interfaces and Service layer. This chapter will describe how the developed system works in live environment.

7.2 Behavior of the MASDSS in live

This section describes the features and functionality of developed system.

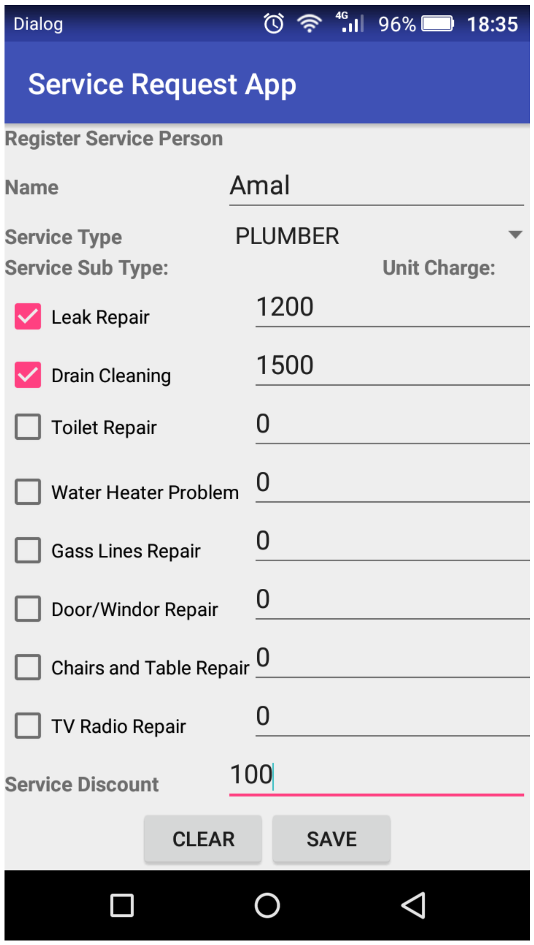


1)



2)

Figure 7.2.1: Sample Mobile User Interfaces



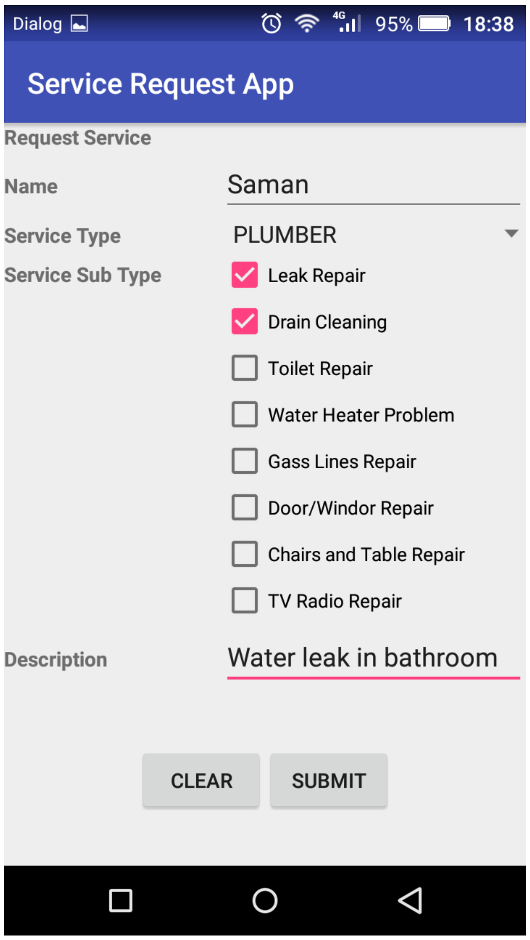
Register Service Person

Name	Amal
Service Type	PLUMBER

Service Sub Type: Unit Charge:

[✓] Leak Repair	1200
[✓] Drain Cleaning	1500
Toilet Repair	0
Water Heater Problem	0
Gass Lines Repair	0
Door/Windor Repair	0
Chairs and Table Repair	0
TV Radio Repair	0

Service Discount 100



Request Service

Name	Saman
Service Type	PLUMBER
Service Sub Type	[✓] Leak Repair
	[✓] Drain Cleaning
	Toilet Repair
	Water Heater Problem
	Gass Lines Repair
	Door/Windor Repair
	Chairs and Table Repair
	TV Radio Repair
Description	Water leak in bathroom

3) 4)

Figure 7.2.2: Sample Mobile User Interfaces

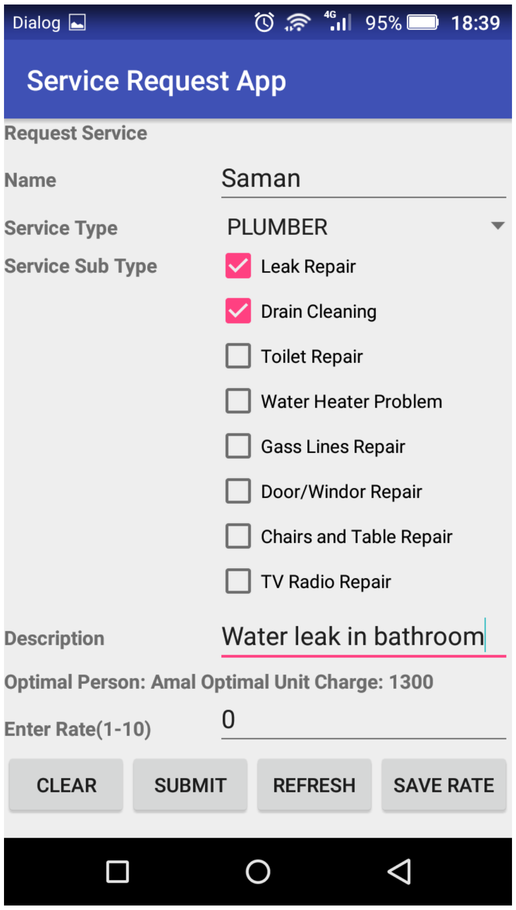
1) This screen is the main UI and user can select below operation,

   - Register service provider :- Enter required service provider information and register in the system as a service provider.

   - Request service :- Any user who has installed this app on his mobile phone can access and request perticular service.

   - Setting :- Required system setting can be set in app using this option.

2) In this screen it shows operations as menu items and users can select one of above operations. Same operations which were described above can be access using this alternative option.



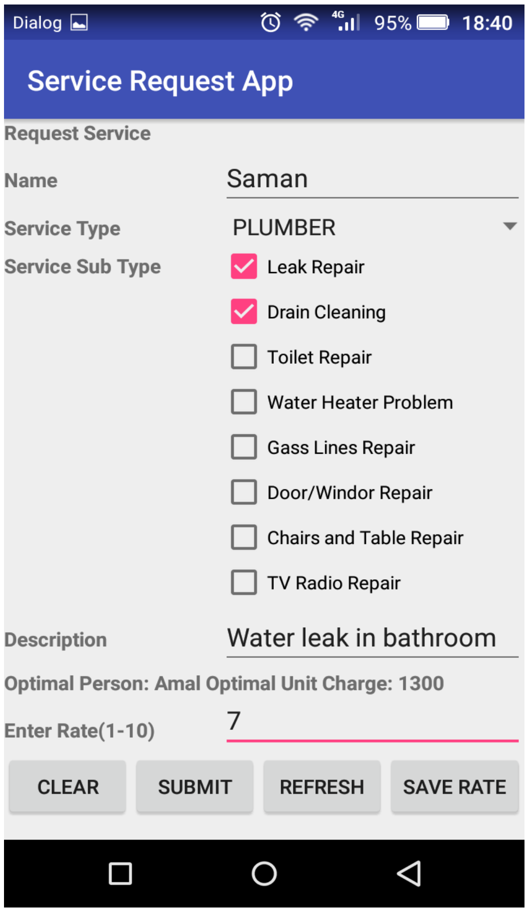
Request Service

Name	Saman
Service Type	PLUMBER
Service Sub Type	[✓ Leak Repair]
	[✓ Drain Cleaning]
	Toilet Repair
	Water Heater Problem
	Gass Lines Repair
	Door/Windor Repair
	Chairs and Table Repair
	TV Radio Repair
Description	Water leak in bathroom

Optimal Person: Amal Optimal Unit Charge: 1300

Enter Rate(1-10)	0

5)



Request Service

Name	Saman
Service Type	PLUMBER
Service Sub Type	[✓ Leak Repair]
	[✓ Drain Cleaning]
	Toilet Repair
	Water Heater Problem
	Gass Lines Repair
	Door/Windor Repair
	Chairs and Table Repair
	TV Radio Repair
Description	Water leak in bathroom

Optimal Person: Amal Optimal Unit Charge: 1300

Enter Rate(1-10)	7

6)

Figure 7.2.3: Sample Mobile User Interfaces

3) In this screen it dispays how to register a service person. There, he has to enter his providing service such as Plumber, Carpenter, Electrician ,Car/Vehicle hire or Car/Vehicle repair and sub service information such as leak repair ,drain cleaning toilet repair ,water heater problem, door window repair or TV Radio repair and unit charge of each sub service and also the maximum discount value which he is offering for the client.

4) Using screen 4 any client can requst a service by entering requird parameters. If he is not aware about service, he can keep it blank and just select sub service types and write his description. This description will help service provider to have idea

about domestic system failure. And his name is important for keeping records of required service.

5) Once client submit his request then algorithm process input parameters and finds required service and optimal service provider and display on the screen. As example shown in screen 5 for the failure request, water leak problem in bathroom made by Saman is served by Amal with unit charge 1300. These information is displayed in client's screen.
6) Once service provider completes the service then he will be rated by the client. In this scenario, service requester can enter a value between 1 and 10 in field 'Enter rate' and save rating value in the system. This rating value will be useful for service provider to have next service in service pool.

7.3 Summary

This section described how system is working in live and its own features. In next chapter it will describe how we evaluate the developed model and application. That section will describe the information about experimental design, Experimental test cases, Evaluation strategy and obtained experimental results.

Chapter 8

Evaluation

8.1 Introduction

Chapter 6 explained about the detail of implementation of our research and this chapter will discussed the final evaluation section of the project. While developing system, incremental evaluation was done and we observed satisfactory results and we were encouraged by those results. In this chapter we describe detailed information about Evaluation strategy, experimental design and results.

8.2 Experimental Design

Experiments were designed to evaluate the accuracy and optimality of the solutions which are obtained from the developed model.

For evaluating the model, various test cases are built and tested and here it describes some test cases. Using mobile UI service providers are registered in the system and then service requesters put requests for particular services. Message space of multi-agent engine is analysed and results are evaluated. Sample data set and evaluation information is mentioned here.

8.3 Evaluation Strategy

For various parameter combinations such as distance between service provider and client agents (different latitude and longitude values), Service type and service subtypes, rating values of service provider agents, unit charge and discount values, different service provider and service requester agents are created in agent container and evaluate the solutions of the model. For this purpose we display created sample data entries in below tables.

Table 01:- In this table it displays parameter values such as providing service information of each service provider.

Table 02:- Sub service information with unit charge information is displayed in this table.

8.4 Experimental Sample Data

NAME	LOCATION LATITUDE	LOCATION LONGITUDE	SERVICE	SERVICE RATING	CHARGE DISCOUNT
Sanjaya	50.23444	-24.222	PLUMBER	2.5	50
Kapila	50.23400	-24.221	PLUMBER	2.3	0
Dilshan	7.302228	79.893934	CARPENTER	1.4	100
Mahinda	7.30222	79.89393	CARPENTER	1.4	90
Nilantha	6.302228	80.893934	Electrician	1.5	80
Niroshan	7.302228	78.8939	Electrician	2	0
Nishantha	6.302228	81.893934	PLUMBER	5.6	0
Kalsha	6.302228	79.893934	PLUMBER	5.6	0

Table 8.2.1. *Parameter information of Service providers*

NAME	SERVICE	DESCRIPTION	UNIT CHARGE
Sanjaya	PLUMBER	Leak Repair	500
Sanjaya	PLUMBER	Drain Cleaning	550
Sanjaya	PLUMBER	Toilet Repair	800
Kapila	PLUMBER	Leak Repair	500
Kapila	PLUMBER	Drain Cleaning	450
Kapila	PLUMBER	Toilet Repair	800
Dilshan	CARPENTER	Chairs and Table Repair	1500
Dilshan	CARPENTER	Door/Window Repair	1200
Mahinda	CARPENTER	Chairs and Table Repair	1320
Mahinda	CARPENTER	Door/Window Repair	1200
Nilantha	ELECTRICIAN	TV/Radio Repair	2000
Niroshan	ELECTRICIAN	TV/Radio Repair	1800
Nishantha	PLUMBER	Leak Repair	2000
Nishantha	PLUMBER	Drain Cleaning	3000
Kalsha	PLUMBER	Leak Repair	2000
Kalsha	PLUMBER	Drain Cleaning	3000

Table 8.2.2. *Details of registered services of corresponding service providers*

8.5 Experimental Results

8.5.1 Test Case 01

In this scenario client knows the service which he should request and he then select required parameters and submit his request.

Generate service requester with below information,
Name = Saman, LOCATION LATITUDE= 50.2310, LOCATION LONGITUDE= -24.2888
Request Service= PLUMBER [Leak Repair, Toilet Repair]

The message space spool output is mentioned in appendix 04.

In this situation according to the multi agent model, requester 'Saman' is served by 'Sanjaya' with unit charge 600.

8.5.2 Test Case 02

In this scenario also client knows the service which he should request and he then select required parameters and submit his request.

Generate service requester with below information,
Name = Meena, LOCATION LATITUDE= 8.3022281, LOCATION LONGITUDE= 76.8939343
Request Service= PLUMBER [Leak Repair, Drain Cleening]

The message space spool output is mentioned in appendix 04.

In this situation according to the multi agent model, requester 'Meena' is served by 'Kalsha' with unit charge 2000.

8.5.3 Test Case 03

In this scenario client is not aware about the service but he knows only the sub service information, He then just select what he experience in domestic failure and submit.

Generate service requester with below information,
Name=Silva, LOCATION LATITUDE= 7.3022281, LOCATION LONGITUDE= 79.89393
Request Service= N/A Sub Service info: [Chairs and Table Repair, Door/Window Repair]
In this situation Model has selected the service as CARPENTER and service provider Dilshan has been selected with unit cost 1290.

The message space spool output is mentioned in appendix 04.

8.5.4 Overall results of executed test cases.

TEST CASE	REQUESTED OR SELECTED SERVICE	OPTIMAL SERVICE PROVIDER	UNIT CHARGE
01	PLUMBER	SANJAYA	600
02	PLUMBER	KALSHA	2000
03	CARPENTER	DILSHAN	1290

Table 8.5.4.1 Overall results

While developing the system incremental evaluation is done and further we intend to evaluate live implemented system by monitoring and analysing below facts,

- How many clients come back to have services(growth rate of client base)
- Customer feedback and customer satisfaction (analyze the rating given by the client after completing the service
- Time taken to complete the service(efficiency of the service)
- Revenue generated in service provider’s side

8.6 Summary

The evaluation section was described in this chapter and various test cases are designed and tested for the evaluation. It explained about experimental design, various test cases, evaluation strategy and experimental results. Instead of this we conducted parallel testing while development was carried out and we observed satisfactory results. In next chapter we will explain the conclusion and further work of our research project.

Chapter 9

Conclusion and Further Work

9.1 Introduction

In the previous chapter we discussed the evaluation section of our research project and it described experimental design evaluation strategy and experimental results of each test cases. And this chapter will concludes the result obtained in previous chapter. And also it will describe any drawbacks and further improvements of the developed model and application.

9.2 Concluding Research and Further Work

When we analyses the experimental results which we obtained from the experimental design and test cases which were described in evaluation section, we can conclude our initial hypothesis that our developed model is giving optimal solutions to find best service person.

This paper presented our novel approach to solve distributed service problem. We have developed a model and software system named as MASDSS using multi agent technology and Jade libraries to have required service, cost and time effectively. Service requesters/clients can access this model through developed android mobile application and enter required service parameters then the model do the process and chose the most suitable service and service provider to serve this client and these information is displayed in the screens of smartphone of two parties. This MASDSS is an intelligent system who can assist even people in domestic environment who are technically non competent or may be disable to solve issues in an emergency situation and also support to have quick decision for even technically competent persons.

In this developed model, it consume mentioned inputs and finds one phase optimized solutions but as further work we can developed this model to find chain of solutions. As example one solution of service provider may be to serve particular client within given time frame and go to serve next client in next allocated time frame. The instructions and directions can be shown on the Google map itself.

Using this software application and algorithm people can have satisfactory cost and time effective services even if they do not know the required service, form most suitable service providers who are distributed over the island. And also service providers can be economically rich by providing satisfactory service.

9.3 Summary

In chapter 1, introduction for the research was explained and aim and objectives were defined for this. Critical review of the developments and issues in the area of distributed service systems were studied and descried in chapter 2 and also defined the research problem and identified required technologies to solve the problem. In chapter 3 technology part of our research were explained and in there Multi Agent technology was used for solving the network and finding the best service provider to serve the requester. In addition to that technology MySQL Database technology, Android architecture, Spring boot and Service architecture were used. The design section of MASDSS was explained in chapter 4 and there, it explained how each processes interact each other and what each process are performing. Database design, Agent Design and Application designs were mainly explained in that section. In chapter 6 we described the main implementation part of our MASDSS, Using MySQL database technology the database of our system was implemented and using Java technology with Jade library main multi agent engine was coded. User interface were developed using Android studio IDE and APIs are developed and implemented using Spring Boot.

The interesting thing is to see how the systems works in live environment and that part was explained in chapter 7 with attractive screen shots of mobile UI. In chapter 8 experimental design and evaluation strategy were defined and experimental results were collected executing defined test cases and that was to evaluate the solutions which was given by the developed model, in addition to that while doing system development we obtained satisfactory results. After analyzing collected results and facts we conclude that our initials hypotheses in current chapter that our developed model is giving optimal satisfactory solution to find best service person.

Reference

[1] M. Ramezani and M. Nourinejad, “Dynamic modeling and control of taxi services in large-scale urban networks: A macroscopic approach,” *Transportation Research Procedia*, vol. 23, pp. 41–60, Jan. 2017.

[2] K. T. Seow, N. H. Dang, and D.-H. Lee, “A Collaborative Multiagent Taxi-Dispatch System,” p. 10.

[3] W. Xian, “Dispatching Strategies for the Taxi-Customer Searching Problem in the Booking Taxi Service,” p. 15, 2013.

[4] M. Tlig and N. Bhouri, “A Multi-Agent System for Urban Traffic and Buses Regularity Control,” *Procedia - Social and Behavioral Sciences*, vol. 20, pp. 896–905, 2011.

[5] J. L. Adler, G. Satapathy, V. Manikonda, B. Bowles, and V. J. Blue, “A multi-agent approach to cooperative traffic management and route guidance,” *Transportation Research Part B: Methodological*, vol. 39, no. 4, pp. 297–318, May 2005.

[6] E. H. Durfee, “Distributed Problem Solving and Multi-Agent Systems: Comparisons and Examples,” p. 11.

[7] P. Davidsson, L. Henesey, L. Ramstedt, J. Törnquist, and F. Wernstedt, “An analysis of agent-based approaches to transport logistics,” *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 4, pp. 255–271, Aug. 2005.

[8] Bo Chen and H. H. Cheng, “A Review of the Applications of Agent Technology in Traffic and Transportation Systems,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 485–497, Jun. 2010.

[9] A. Namoun, C. A. Marín, B. Saint Germain, N. Mehandjiev, and J. Philips, “A multi-agent system for modelling urban transport infrastructure using intelligent traffic forecasts,” in *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, 2013, pp. 175–186.

[10] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, “Conflict-based search for optimal multi-agent pathfinding,” *Artificial Intelligence*, vol. 219, pp. 40–66, Feb. 2015.

[11] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, “The increasing cost tree search for optimal multi-agent pathfinding,” *Artificial Intelligence*, vol. 195, pp. 470–495, Feb. 2013.

[12] J. FAN, Z. GUO, and H. Yuan, “Research on Improved Dijkstra Algorithm using for Safety Management in the Road Network.”

[13] M. Gath, O. Herzog, and M. Vaske, “Parallel Shortest-path Searches in Multiagent-based Simulations with PlaSMA:,” 2015, pp. 15–21.

[14] S. TAO, “Mobile Phone-based Vehicle Positioning and Tracking and Its Application in Urban Traffic State Estimation,” p. 99.

[15] N. Ahmadullah, S. Islam, and T. Ahmed, *RouteFinder: Real-time Optimum Vehicle Routing using Mobile Phone Network*. .

[16] K. Hager, J. Rauh, and W. Rid, “Agent-based Modeling of Traffic Behavior in Growing Metropolitan Areas,” *Transportation Research Procedia*, vol. 10, pp. 306–315, 2015.

[17] L. E. Henesey, “Multi-agent systems for container terminal management,” Blekinge Institute of Technology, Karlskrona, 2006.

[18] “Determinants of Route Choice and the Value of Traveler Information.” .

[19] "Dynamic routing using the network of car drivers." .

[20] B. Chen, H. H. Cheng, and J. Palen, "Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems," *Transportation Research Part C: Emerging Technologies*, vol. 17, no. 1, pp. 1–10, Feb. 2009.

[21] F. Derakhshan and N. Shahpasandi, "Design and implementation of an urban traffic control system for public transport using multi-agent systems," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 16, no. 10, p. 70, 2016.

[22] A. M. Al-Bakry, "Finding the Best Path Routing Using Multi-agent System," p. 15, 2014.

[23] L. S. Arigela, P. A. Veerendra, S. Anvesh, and K. S. Satya, "Mobile Phone Tracking & Positioning Techniques," vol. 2, no. 4, p. 8, 2013.

[24] S. Cheng, X. Qu, S. Cheng, and X. Qu, "A service choice model for optimizing taxi service delivery," in *in: ITSC'09*, 2009, pp. 1–6.

[25] I. Cobeanu and V. Comnac, "Multi-Agent Systems: Traffic Control Application," *Bulletin of the Transilvania University of Braşov*, vol. 4, no. 53, pp. 107–114, 2011.

[26] J. M. Corchado, D. I. Tapia, and J. Bajo, "A Multi-Agent Architecture for Distributed Services and Applications," p. 32.

[27] M. Guériau, R. Billot, N.-E. El Faouzi, S. Hassas, and F. Armetta, "Multi-Agent Dynamic Coupling for Cooperative Vehicles Modeling.," in *AAAI*, 2015, pp. 4276–4277.

[28] B. Predic, D. Rancic, and A. Milosavljevic, "Impacts of Applying Automated Vehicle Location Systems to Public Bus Transport Management," *Journal of Research and Practice in Information Technology*, vol. 42, no. 2, p. 20, 2010.

[29] I. J. Timóteo, M. R. Araújo, R. J. Rossetti, and E. C. Oliveira, "TraSMAPI: An API oriented towards Multi-Agent Systems real-time interaction with multiple Traffic Simulators," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, 2010, pp. 1183–1188.

[30] V. Wieser, J. Duha, M. Bahleda, R. Odrobiňák, and D. Grendár, "Application of Positioning Systems in Intelligent Transport Systems," Jan. 2003.

[31] Xiao-Feng Xie and Jiming Liu, "Multiagent Optimization System for Solving the Traveling Salesman Problem (TSP)," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 489–502, Apr. 2009.

[32] Z. Liao, "Taxi dispatching via Global Positioning Systems," *IEEE Transactions on Engineering Management*, vol. 48, no. 3, pp. 342–347, Aug. 2001.

[33] R. S Jayant and R. Kumar, "3G MOBILE PHONES POSITIONING SYSTEMS," *Telecommunications*, vol. 57, pp. 67–79, Apr. 2007.

[34] A. Elmahalawy, "INTELLIGENT AGENT AND MULTI AGENT SYSTEMS," *Journal of Engineering and Technology*, vol. 2, Jan. 2012.

[35] "Contract Net Protocol for Coordination in Multi-Agent System," *ResearchGate*. [Online]. Available: https://www.researchgate.net/publication/232636898_Contract_Net_Protocol_for_Coordination_in_Multi-Agent_System. [Accessed: 27-Jan-2019].

Appendices

Appendix 01:

Sample source code segment of manager agent

This manager agent will initialize other agents and maintain and manage the life cycle of other agents.

```
package distributedService;
import jade.core.Agent;
import jade.core.Profile;
import jade.core.ProfileImpl;
import jade.core.Runtime;
import jade.core.behaviours.*;
import jade.wrapper.ContainerController;

public class ManagerAgent extends Agent {

    // The name of the service to be requested
    String argFromMain;
    ContainerController agentContainer;

    protected void setup() {
        // Printout a welcome message
        System.out.println("Hallo! Manager Agent " + getAID().getName() + " is ready.......................");

        Object[] args = getArguments();
        if (args != null && args.length > 0) {
            argFromMain = (String) args[0];
            System.out.println("Arg From Main class " + argFromMain);
            // Add a TickerBehaviour that schedules a request to service provider agents every 8 seconds
            addBehaviour(new TickerBehaviour(this, 10) {
                protected void onTick() {
                    // Relevent agents creation starts
                    createAgents();
                }
            });
        } else {
            // Make the agent terminate
            System.out.println("No target service specified..................................................");
            doDelete();
        }
    }

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

    public void createAgents() {
        try {
            /*
             Create Service provider and service requester agents here
             - Get the service requester's location (from where service request is comming)
             - Find nearest service providers info accroding the defined parameters
             - Initialize relevent service providers accordinly
             - Then Create service requester agent and initialize
             */

            /*Get DB Connection*/
            Long vCont_Id = System.currentTimeMillis();
            String vRequestedService;

            Connection myconnection = ConnectionManager.getConnection();
```

```
	PreparedStatement queryCnt = myconnection.prepareStatement(" select count(*) from distributedservicedb.userpool A where A.status='A' and LoggingAs = 'REQUESTER'"
+ "                  and exists (select * from distributedservicedb.userpool B,distributedservicedb.userpoolsub C where B.status='A' And C.status='A' And"
+ "                             B.UserPool_ID = C.UserPool_UserPool_ID and LoggingAs = 'SERVICEPROVIDER' and B.Current_Service =A.Current_Service) or  (A.Current_Service ='' and A.status='A') or  (A.Current_Service is null and A.status='A')");

/* PreparedStatement queryR = myconnection.prepareStatement("select UserPool_ID,CurrentLocationLan,CurrentLocationLong,KnowService,Current_Service,User_Service_Rating,Mobility,TrafficValue,TimeMatter,status,User_First_Name,Service_Charge_Discount ,LoggingAs from distributedservicedb.userpool where status='A'"
+ "and LoggingAs = 'REQUESTER'");*/

PreparedStatement queryR = myconnection.prepareStatement(" select A.UserPool_ID,A.CurrentLocationLan,A.CurrentLocationLong,A.KnowService,A.Current_Service,A.User_Service_Rating,A.Mobility,A.TrafficValue,A.TimeMatter,A.status,A.User_First_Name,A.Service_Charge_Discount ,A.LoggingAs "
+ " from distributedservicedb.userpool A where A.status='A' and LoggingAs = 'REQUESTER'"
+ "                  and exists (select * from distributedservicedb.userpool B,distributedservicedb.userpoolsub C where B.status='A' And C.status='A' And"
+ "                             B.UserPool_ID = C.UserPool_UserPool_ID and LoggingAs = 'SERVICEPROVIDER' and B.Current_Service =A.Current_Service) or  (A.Current_Service ='' and A.status='A') or  (A.Current_Service is null and A.status='A')  ");


PreparedStatement queryS = myconnection.prepareStatement("select UserPool_ID,CurrentLocationLan,CurrentLocationLong,KnowService,Current_Service,User_Service_Rating,Mobility,TrafficValue,TimeMatter,   status,Service_Charge_Discount,User_First_Name, LoggingAs from distributedservicedb.userpool where status='A'"
+ "and LoggingAs = 'SERVICEPROVIDER' and Current_Service =?");

PreparedStatement querySSub = myconnection.prepareStatement("  select Distinct UserPool_ID,CurrentLocationLan,CurrentLocationLong,KnowService,Current_Service,User_Service_Rating,Mobility,TrafficValue,TimeMatter,   MS.status,Service_Charge_Discount,User_First_Name,LoggingAs "
+ " from distributedservicedb.userpool MS,distributedservicedb.userpoolSUB MSUB"
+ " where  MSUB.UserPool_UserPool_ID = MS.UserPool_ID"
+ " and MS.status='A' "
+ "and MS.LoggingAs = 'SERVICEPROVIDER' "
+ " and exists ( Select * From ( select Sub_Cat_code "
+ "                 from distributedservicedb.userpoolSUB B,distributedservicedb.jobcategory C,distributedservicedb.jobSUBcategory D"
+ "                              Where  B.JobCatid = C.Cat_id And B.JobSubCatId = D.SubCatId And  B.status='A' and C.status='A'"
+ "                              And B.UserPool_UserPool_ID = ? "
+ "                              and B.JobCatid  is not null "
+ "                                                                    union "
+ "            select Sub_Cat_code "
+ "             from distributedservicedb.userpoolSUB B,distributedservicedb.jobSUBcategory D "
+ "                           Where  B.JobSubCatId = D.SubCatId And  B.status='A' and D.status='A' "
+ "                And B.UserPool_UserPool_ID = ? "
+ "                and B.JobCatid  is null) XX "
+ "               Where XX.Sub_Cat_code = MSUB.JobSubCatId )");


//PreparedStatement queryU = myconnection.prepareStatement("update distributedservicedb.userpool set status='X',Agent_Container_Id= " + vCont_Id + " where UserPool_ID = ?");

PreparedStatement queryU = myconnection.prepareStatement("update distributedservicedb.userpool set status='X' where UserPool_ID = ?");


ResultSet rsCnt = queryCnt.executeQuery();
while (rsCnt.next()) {
if (rsCnt.getInt(1) == 0) {
/*.out.println("=====================================================================");
System.out.println("                                                        ");
System.out.println("Currently there are no service requesters in the system");
*/
} else {
/*System.out.println("=====================================================================");
System.out.println("                                                        ");
System.out.println("There are service requesters in the system");
*/
ResultSet rs = queryR.executeQuery();
```

```
        if (agentContainer != null) {
agentContainer.kill();
}

jade.core.Runtime rt = jade.core.Runtime.instance();
Profile p = new ProfileImpl();
//ContainerController agentContainer = rt.createAgentContainer(p);
agentContainer = rt.createAgentContainer(p);



while (rs.next()) {
vRequestedService = rs.getString(5);

ResultSet rsS;
if (vRequestedService == null || vRequestedService.isEmpty()) {
querySSub.setString(1, rs.getString(1));
querySSub.setString(2, rs.getString(1));
rsS = querySSub.executeQuery();
} else {
queryS.setString(1, vRequestedService);
rsS = queryS.executeQuery();
}
while (rsS.next()) {
//queryU.setString(1, rsS.getString(1));
//queryU.executeUpdate();
agentContainer.createNewAgent("S@" + rsS.getString(1) + "@" + rsS.getString(12), "distributedService.ServiceProviderAgent",
new Object[]{rsS.getString(1), rsS.getString(2), rsS.getString(3), rsS.getString(4), rsS.getString(5), rsS.getString(6), rsS.getString(7),
rsS.getString(8), rsS.getString(9), rsS.getString(10), rsS.getString(11),vCont_Id}).start();
}


agentContainer.createNewAgent("R@" + rs.getString(1) + "@" + rs.getString(11), "distributedService.ServiceRequesterAgent",
new Object[]{rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getString(5), rs.getString(6), rs.getString(7), rs.getString(8),
rs.getString(9), rs.getString(10), vCont_Id}).start();

queryU.setString(1, rs.getString(1));
queryU.executeUpdate();


}
}
}

myconnection.close();

} catch (Exception e) {
System.out.println("Error in manager agent.....+++++++++++++++++++++++++++++++++++++++++++++++++");
e.printStackTrace();
}}}
```

Sample source code segment of service provider agent

This source code will implement the service provider agent and this will bind all parameters (service type information of service provider agent) to message body and communicates with requester agent. This agent will execute in separate java thread and uses cyclic behavior for the execution and use ACL standards for the communication.

```
/**
 * ***************************************************************
 * @author Amal Hapuarachchi
 * ***************************************************************
 */
package distributedService;

import jade.core.Agent;
import jade.core.behaviours.*;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import jade.domain.DFService;
import jade.domain.FIPAException;
import jade.domain.FIPANames;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.*;

public class ServiceProviderAgent extends Agent {

   // The list of services (maps the service and its unitCharge)
   private Hashtable serviceList;
   private Hashtable rServiceList;
   // private Map<String, Object> serviceProvidList;
   private Hashtable serviceProvidList;
   private Hashtable mServiceProvidList;
   private String sService;
   private double unitChargeDiscount;
   private double sRating;
   private String sLatitude;
   private String sLongitude;
   private String sStatus;
   private Long sContainerId;
   private String vPoolId;
   private String sTimeMatter;
   private String sKnowService;
   private String sMobility;
   private double sTraficValue;
   private String sRatingString;

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

      try {
         // Create the service list
         serviceList = new Hashtable();
         //serviceProvidList = new HashMap<>();
         serviceProvidList = new Hashtable();
         mServiceProvidList = new Hashtable();

         Connection myconnection = ConnectionManager.getConnection();
         PreparedStatement querySubService = myconnection.prepareStatement("select B.UserPoolSub_ID,Cat_Description ,Sub_Cat_code,Sub_Cat_Description,UntiCharge ,B.UserPool_UserPool_ID "
               + " from distributedservicedb.userpoolSUB B,distributedservicedb.jobcategory C,distributedservicedb.jobSUBcategory D"
               + " Where  B.JobCatid = C.Cat_id And B.JobSubCatId = D.SubCatId And  B.status='A' and C.status='A'"
               + " And B.UserPool_UserPool_ID = ?");
         Object[] args = getArguments();
         if (args != null && args.length > 0) {
            //unitCharge = Double.parseDouble((String) args[1]);
            vPoolId = (String) args[0];
            sLatitude = (String) args[1];
            sLongitude = (String) args[2];
            sKnowService = (String) args[3];
            sService = (String) args[4];
            sRating = Double.parseDouble((String) args[5]);
            sMobility = (String) args[6];
            sTraficValue = Double.parseDouble((String) args[7]);
            sTimeMatter = (String) args[8];
            sStatus = (String) args[9];
            unitChargeDiscount= Double.parseDouble((String) args[10]);
            sContainerId = (Long) args[11];

            querySubService.setString(1, vPoolId);
            ResultSet rsSubService = querySubService.executeQuery();
            while (rsSubService.next()) {
               serviceProvidList.put(rsSubService.getString(1),//Key Value
                        sService + "@"//0
                        + sLatitude + "@"//1
                        + sLongitude + "@"//2
                        + sMobility + "@"//3
                        + sTraficValue + "@"//4
                        + sRating + "@"//5
                        + sTimeMatter + "@"//6
                        + sStatus + "@"//7
                        + rsSubService.getString(2) + "@"//8
                        + rsSubService.getString(3) + "@"//9
                        + rsSubService.getString(4) + "@"//10
                        + rsSubService.getString(5) + "@"//11
                        + unitChargeDiscount);//12
            }

            this.updateserviceList(serviceProvidList);
         }

         myconnection.close();
      } catch (Exception e) {
         e.printStackTrace();
      }
      // Register the services in the yellow pages
      DFAgentDescription dfd = new DFAgentDescription();
      dfd.setName(getAID());
      ServiceDescription sd = new ServiceDescription();

      sd.setType("service-providing");
      sd.setName("JADE-service-providing");
      dfd.addServices(sd);
      try {
         DFService.register(this, dfd);
      } catch (FIPAException fe) {
         System.out.println("Exception 02");
         fe.printStackTrace();
      }

      // Add the behaviour serving queries from requester agents
      addBehaviour(new OfferRequestsServer());

      // Add the behaviour serving requests from requester agents
      addBehaviour(new PurchaseOrdersServer());
   }

   // Put agent clean-up operations here
   protected void takeDown() {
      // Deregister from the yellow pages
      try {
         DFService.deregister(this);
      } catch (FIPAException fe) {
         System.out.println("Exception 03");
         fe.printStackTrace();
      }
      // Close the GUI
      // myGui.dispose();
```

```
    // Printout a dismissal message
    System.out.println("Service Provider Agent " + getAID().getName() + " terminating................................");
    System.out.println("                                                                    ");

}

/**
 * This is invoked by the GUI when the user adds a new service
 */
public void updateserviceList(final Hashtable pServiceProvidList) {
    addBehaviour(new OneShotBehaviour() {
        public void action() {
            serviceList = pServiceProvidList;
            /* Object[] keys = serviceList.keySet().toArray();
             for (Object key : keys) {
             System.out.println(getAID().getName() + " service Provider list Key: " + key + "; Value: " + serviceList.get(key));
             }*/
        }
    });
}

private class OfferRequestsServer extends CyclicBehaviour {

    public void action() {
        try {
               int repliesCnt = 0;

            String rServiceString = null;
            String sServiceString = null;
            String smServiceString = null;
            int subServiceMatchings = 0;

            double unitChargeDiscount;
            double unitCharge;
            Random randomGenerator ;
                            int amtDiscount;
          //  MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.CFP);

            MessageTemplate mt = MessageTemplate.and(
                                                        MessageTemplate.MatchProtocol(FIPANames.InteractionProtocol.FIPA_CONTRACT_NET),
                                                        MessageTemplate.MatchPerformative(ACLMessage.CFP) );

            ACLMessage msg = myAgent.receive(mt);

            if (msg != null) {

              if (Integer.parseInt(msg.getOntology())>0)
              {

                             Object[] sMKeys = serviceList.keySet().toArray();
                     for (Object key : sMKeys) {
                         smServiceString = serviceList.get(key).toString();
                         String[] sMServiceStringList = smServiceString.split("@", 13);
                        // System.out.println("sMServiceStringList amnt ::" + sMServiceStringList[11]);
                        // System.out.println("sMServiceStringList discont ::" + sMServiceStringList[12]);

                         unitCharge = Double.parseDouble((String) sMServiceStringList[11]);
                         unitChargeDiscount = Double.parseDouble((String) sMServiceStringList[12]);

                             randomGenerator = new Random();
                                      amtDiscount = randomGenerator.nextInt(10) * 10;

                                      System.out.println("Current Unit Charge ::" + unitCharge);
                                      System.out.println("Discount Amount::" + amtDiscount);

                                     if (amtDiscount>=unitChargeDiscount)
                                              amtDiscount = (int) unitChargeDiscount;
```

```
                unitCharge= unitCharge - amtDiscount;
                System.out.println("New unit Charge ::" + unitCharge);

               mServiceProvidList.put(key,//Key Value
                                    sMServiceStringList[0] + "@"//0
                    + sMServiceStringList[1] + "@"//1
                    + sMServiceStringList[2] + "@"//2
                    + sMServiceStringList[3] + "@"//3
                    + sMServiceStringList[4] + "@"//4
                    + sMServiceStringList[5] + "@"//5
                    + sMServiceStringList[6] + "@"//6
                    + sMServiceStringList[7] + "@"//7
                    + sMServiceStringList[8] + "@"//8
                    + sMServiceStringList[9] + "@"//9
                    + sMServiceStringList[10] + "@"//10
                    + unitCharge + "@"//11
                    + sMServiceStringList[12]);//12
            }

              // serviceList.clear();
               serviceList = mServiceProvidList;
        }

         // CFP Message received. Process it
         // ------------Service Requester's info------------------------
         Hashtable rServiceList = (Hashtable) msg.getContentObject();
         Object[] rKeys = rServiceList.keySet().toArray();
         for (Object key : rKeys) {
             rServiceString = rServiceList.get(key).toString();
            // System.out.println("rServiceString ::" + rServiceString);
         }

         String[] rServiceStringList = rServiceString.split("@", 12);
         String rService = rServiceStringList[0];
         String rStatus = rServiceStringList[7];
         // ------------------------------------------------------
         // ------------Service Provider's info------------------------
        Object[] sKeys = serviceList.keySet().toArray();
         for (Object key : sKeys) {
             sServiceString = serviceList.get(key).toString();
            //System.out.println("sServiceString ::" + sServiceString);
            // System.out.println("--------------------------------------------------------");
         }

         String[] sServiceStringList = sServiceString.split("@", 13);
         String sService = sServiceStringList[0];
         String sStatus = sServiceStringList[7];
         String service = sService;
        // ------------------------------------------------------
         ACLMessage reply = msg.createReply();
         if (rService == null || rService.isEmpty() || rService.equalsIgnoreCase("null")) {
             rKeys = rServiceList.keySet().toArray();
             for (Object rkey : rKeys) { // ------------Service Provider's info----------------------
                 rServiceString = rServiceList.get(rkey).toString();
                 rServiceStringList = rServiceString.split("@", 12);
                //PLUMBER@1222773@22883232@null@131.0@5.6@null@A@PLUMBER@4@Garbage                          Disposal
Repair@1500@2000
                 sKeys = serviceList.keySet().toArray();
                 for (Object skey : sKeys) { // ------------Service Provider's info----------------------
                     sServiceString = serviceList.get(skey).toString();
                     System.out.println("sServiceString ::" + sServiceString);
                     sServiceStringList = sServiceString.split("@", 13);

                     if (sServiceStringList[9].equals(rServiceStringList[9])) {//check sub service
                         subServiceMatchings++;
                     }
                    }
                }
                if (subServiceMatchings > 0) {
                        repliesCnt++;
                    reply.setPerformative(ACLMessage.PROPOSE);
                    // reply.setContent(paraList);
                    reply.setContentObject(serviceList);
                }
            } else {
                if (rService.equals(service) && sStatus.equals("A")) {
                    // The requested service is available. Reply with the full service info
                    reply.setPerformative(ACLMessage.PROPOSE);
                    // reply.setContent(paraList);
                    reply.setContentObject(serviceList);

                } else {
                    // The requested service is NOT available.
                    reply.setPerformative(ACLMessage.REFUSE);
                    reply.setContent("not-available");
                    // System.out.println(" not-available");
                }
            }
            myAgent.send(reply);
        } else {
            block();
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
  }
} // End of inner class OfferRequestsServer

private class PurchaseOrdersServer extends CyclicBehaviour {

  public void action() {

    try {
        String rServiceString = null;
        String sServiceString = null;
        int subServiceMatchings = 0;
        MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.ACCEPT_PROPOSAL);
        ACLMessage msg = myAgent.receive(mt);
        if (msg != null) {
        // ACCEPT_PROPOSAL Message received. Process it
            // String title = msg.getContent();

            // ------------Service Requester's info------------------------
            Hashtable rServiceList = (Hashtable) msg.getContentObject();
            Object[] rKeys = rServiceList.keySet().toArray();
            for (Object key : rKeys) {
                rServiceString = rServiceList.get(key).toString();

            }

            String[] rServiceStringList = rServiceString.split("@", 12);
            String rService = rServiceStringList[0];
            String rStatus = rServiceStringList[7];
            // ------------------------------------------------------

            // ------------Service Provider's info------------------------
            Object[] sKeys = serviceList.keySet().toArray();
            for (Object key : sKeys) {
                sServiceString = serviceList.get(key).toString();
            }
```

```
                String[] sServiceStringList = sServiceString.split("@", 12);
                String sService = sServiceStringList[0];
                String sStatus = sServiceStringList[7];
                String service = sService;
                // -------------------------------------------------------
                // String rService = msg.getContent();
                ACLMessage reply = msg.createReply();

                if (rService == null || rService.isEmpty() || rService.equalsIgnoreCase("null")) {
                    rKeys = rServiceList.keySet().toArray();
                    for (Object rkey : rKeys) { // ------------Service Provider's info-----------------------
                        rServiceString = rServiceList.get(rkey).toString();
                        rServiceStringList = rServiceString.split("@", 12);
                        //PLUMBER@1222773@22883232@null@131.0@5.6@null@A@PLUMBER@4@Garbage                    Disposal
Repair@1500@2000
                        sKeys = serviceList.keySet().toArray();
                        for (Object skey : sKeys) { // ------------Service Provider's info-----------------------
                            sServiceString = serviceList.get(skey).toString();
                            sServiceStringList = sServiceString.split("@", 12);

                            if (sServiceStringList[9].equals(rServiceStringList[9])) {//check sub service
                                subServiceMatchings++;
                            }

                        }
                    }
                    if (subServiceMatchings > 0) {

                        reply.setPerformative(ACLMessage.INFORM);

                        myAgent.doDelete();
                    }

                } else {
                    if (rService.equals(service) && sStatus.equals("A")) {
                        reply.setPerformative(ACLMessage.INFORM);
                        //serviceList.put("status", "E");
                        System.out.println("=======================================================================");
                        System.out.println("                                                        ");
                        System.out.println("service " + rService + " is provided to agent " + msg.getSender().getName());
                        myAgent.doDelete();
                    } else {
                            // The requested service has been issues to another service
                        // requester in the meanwhile .
                        reply.setPerformative(ACLMessage.FAILURE);
                        reply.setContent("not-available");
                    }
                }
                myAgent.send(reply);
            } else {
                block();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
  } // End of inner class OfferRequestsServer}
```

Sample source code segment of service requester agent

This java class implements the behavior of service requester agent it will broadcast service request information binding to message body to service provider agents in agent container. This agent will execute in separate java thread and uses cyclic behavior for the execution and use ACL standards for the communication.

```
/**
 * ****************************************************************
 * @author Amal Hapuarachchi
 * *****************************************************************
 */
package distributedService;
import jade.core.Agent;
import jade.core.AID;
import jade.core.behaviours.*;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import jade.domain.DFService;
import jade.domain.FIPAException;
import jade.domain.FIPANames;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.proto.ContractNetInitiator;

public class ServiceRequesterAgent extends Agent {
    // The name of the service to be requested
    private String targetService;
    // The list of known service provider agents
    private AID[] serviceProviderAgents;
    private String rLatitude;
    private String rLongitude;
    private Long rContainerId;
    private double rRating;
    private String rTimeMatter;
    private String rKnowService;
    private String rMobility;
    private double rTraficValue;
    private String rStatus;
    private Hashtable serviceReqList;
    private Hashtable serviceProvidngList;
    private ServiceProviderAgent serviceProviderAgent;
    private String vPoolId;

    // Put agent initializations here
    protected void setup() {
        try {
            // Printout a welcome message
            System.out.println("Hallo! Service requester-agent " + getAID().getName() + " is ready.");

            //serviceReqList = new HashMap<>();
            serviceReqList = new Hashtable();

            Connection myconnection = ConnectionManager.getConnection();
            PreparedStatement querySubService = myconnection.prepareStatement("select B.UserPoolSub_ID ,Cat_Description
,Sub_Cat_code,Sub_Cat_Description,UntiCharge,B.UserPool_UserPool_ID "
```

```
                + "                    from distributedservicedb.userpoolSUB B,distributedservicedb.jobcategory
C,distributedservicedb.jobSUBcategory D"
                + "                    Where  B.JobCatid = C.Cat_id And B.JobSubCatId = D.SubCatId And  B.status='A' and C.status='A'"
                + "                    And B.UserPool_UserPool_ID = ?"
                + "                    and B.JobCatid  is not null"
                + "                    union"
                + "                              select B.UserPoolSub_ID ,null Cat_Description
,Sub_Cat_code,Sub_Cat_Description,UntiCharge,B.UserPool_UserPool_ID "
                + "                              from distributedservicedb.userpoolSUB B,distributedservicedb.jobSUBcategory D"
                + "                              Where  B.JobSubCatId = D.SubCatId And  B.status='A' and D.status='A'"
                + "     And B.UserPool_UserPool_ID = ?"
                + "     and B.JobCatid  is null");

            // Get the name of the service to provide as a start-up argument
            Object[] args = getArguments();
            if (args != null && args.length > 0) {
                vPoolId = (String) args[0];
                rLatitude = (String) args[1];
                rLongitude = (String) args[2];
                rKnowService = (String) args[3];
                targetService = (String) args[4];
                rRating = Double.parseDouble((String) args[5]);
                rMobility = (String) args[6];
                rTraficValue = Double.parseDouble((String) args[7]);
                rTimeMatter = (String) args[8];
                rStatus = (String) args[9];
                rContainerId = (Long) args[10];

                querySubService.setString(1, vPoolId);
                querySubService.setString(2, vPoolId);
                ResultSet rsSubService = querySubService.executeQuery();
                while (rsSubService.next()) {
                    serviceReqList.put(rsSubService.getString(1),//Key Value
                            targetService + "@"//0
                            + rLatitude + "@"//1
                            + rLongitude + "@"//2
                            + rMobility + "@"//3
                            + rTraficValue + "@"//4
                            + rRating + "@"//5
                            + rTimeMatter + "@"//6
                            + rStatus + "@"//7
                            + rsSubService.getString(2) + "@" +//Cat_Description//8
                            rsSubService.getString(3) + "@" + //Sub_Cat_code//9
                            rsSubService.getString(4) + "@" +//Sub_Cat_Description//10
                            rsSubService.getString(5) //UntiCharge//11
                    );
                }


                // Add a TickerBehaviour that schedules a request to service provider agents every 8 seconds
                addBehaviour(new TickerBehaviour(this, 100) {
                    protected void onTick() {
                        System.out.println("Trying to serve " + targetService);
                        System.out.println("======================================================================");
                        System.out.println("                                                             ");
                        // Update the list of service provider agents
                        DFAgentDescription template = new DFAgentDescription();
                        ServiceDescription sd = new ServiceDescription();
                        sd.setType("service-providing");
                        template.addServices(sd);
                        try {
                            DFAgentDescription[] result = DFService.search(myAgent, template);
                            System.out.println("Found the following service provider agents:");
                            System.out.println("========================================================================");
                            System.out.println("                                                              ");
                            serviceProviderAgents = new AID[result.length];
                            for (int i = 0; i < result.length; ++i) {
                                serviceProviderAgents[i] = result[i].getName();
                                System.out.println(serviceProviderAgents[i].getName());
                            }
                        } catch (FIPAException fe) {
                            System.out.println("Exception 01");
                            fe.printStackTrace();
                        }

                        // Perform the request
                        myAgent.addBehaviour(new RequestPerformer());
                    }
                });
            } else {
                // Make the agent terminate
                System.out.println("No target service specified");
                System.out.println("                                                              ");
                doDelete();
            }

            myconnection.close();

          } catch (Exception e) {
            e.printStackTrace();
          }
        }

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

        private class RequestPerformer extends Behaviour {

          private AID bestServiceProvider; // The agent who provides the best offer
          private double bestUnitCharge;  // The best offered unitCharge
          private double bestRate;  // The best rate of the service provider agent
          private double bestDistance;  // The minimum distance between service provider and the request
          private String bestLocation;  //
          private String bestService;  //
          private int bestNoOfSubServieMatch;  //
          private double bestOptimizeVal;
          private String statusActive;  // The service provider is in active status
          private int repliesCnt = 0; // The counter of replies from Service provider agents
          private MessageTemplate mt; // The template to receive replies
          private String sOptiServiceProvider;
          private String sOptiUserPoolID;
          private String[] sAgentInfo;
        private int step = 0;
        double avgUnitCharge;
        double distance;
        double optimizeVal;
        private int changeableCnt = 1;
        private int changedCnt = 0;

        public void action() {

          switch (step) {
            case 0:
              // Send the cfp to all Service providers
              ACLMessage cfp = new ACLMessage(ACLMessage.CFP);
              for (int i = 0; i < serviceProviderAgents.length; ++i) {
                cfp.addReceiver(serviceProviderAgents[i]);
              }
              try {
                // System.out.println(" service requester step 0");
                cfp.setContentObject(serviceReqList);
                cfp.setConversationId("service-trade");
                cfp.setProtocol(FIPANames.InteractionProtocol.FIPA_CONTRACT_NET); //***setting protocol*/
                cfp.setReplyWith("cfp" + System.currentTimeMillis()); // Unique value
                cfp.setOntology(Integer.toString(changedCnt));
                myAgent.send(cfp);
                // Prepare the template to get proposals
                mt = MessageTemplate.and(MessageTemplate.MatchConversationId("service-trade"),
                    MessageTemplate.MatchInReplyTo(cfp.getReplyWith()));
                step = 1;
                break;
              } catch (Exception e) {
                e.printStackTrace();
              }
            case 1:
              // Receive all proposals/refusals from seller agents
              ACLMessage reply = myAgent.receive(mt);
              if (reply != null) {
                // Reply received
                if (reply.getPerformative() == ACLMessage.PROPOSE) {

                  try {
                                      // This is an offer

                    //System.out.println(" service requester step 1 PROPOSE");
                    serviceProvidngList = (Hashtable) reply.getContentObject();
                    double traficCondition = rTraficValue;
                    String rServiceString = null;
                    String[] rServiceStringList;

                    double sLatitude = 0 ;
                    double sLongitude = 0 ;
                    double rLatitude = 0 ;
                    double rLongitude= 0 ;
                    String sServiceString = null;
                    int subServiceMatchings = 0;
                    double sumUnitCharge = 0;
                    double sRating = 0;
                    String[] sServiceStringList;
                Object[] rKeys = serviceReqList.keySet().toArray();
                for (Object rkey : rKeys) { // ------------Service Provider's info-----------------------
                  rServiceString = serviceReqList.get(rkey).toString();
                  rServiceStringList = rServiceString.split("@", 12);

                  rLatitude = Double.parseDouble((String) rServiceStringList[1]);
                  rLongitude = Double.parseDouble((String) rServiceStringList[2]);

                  //PLUMBER@1222773@22883232@null@131.0@5.6@null@A@PLUMBER@4@Garbage Disposal
        Repair@1500@2000
                  Object[] sKeys = serviceProvidngList.keySet().toArray();
                  for (Object skey : sKeys) { // ------------Service Provider's info-----------------------
                    sServiceString = serviceProvidngList.get(skey).toString();
                    sServiceStringList = sServiceString.split("@", 13);

                    sLatitude = Double.parseDouble((String) sServiceStringList[1]);
                    sLongitude = Double.parseDouble((String) sServiceStringList[2]);
                    sRating = Double.parseDouble((String) sServiceStringList[5]);

                    if (sServiceStringList[0].equals(rServiceStringList[0])) {//check service
                         bestService = sServiceStringList[0];
                    }

                    if (sServiceStringList[9].equals(rServiceStringList[9])) {//check sub service
                        //System.out.println("check sub service r Val = " + rServiceStringList[9]);
                        // System.out.println("check sub service s Val = " + sServiceStringList[9]);
                      subServiceMatchings++;
                      sumUnitCharge = sumUnitCharge + Double.parseDouble((String) sServiceStringList[11]);
                      bestService = sServiceStringList[0];
                    }

                  }
                }
                if (subServiceMatchings == 0) subServiceMatchings = 1;
                if (traficCondition == 0) traficCondition = 1;

                avgUnitCharge = sumUnitCharge / subServiceMatchings;

                distance = DistanceCalculator.getDistance(sLatitude, sLongitude, rLatitude, rLongitude, "K");
                if (distance==0) distance=1;

                optimizeVal = (sRating * subServiceMatchings) / (distance * traficCondition * avgUnitCharge);
                System.out.println("                                                              ");
                System.out.println("Service Provider Agent = " + reply.getSender().getLocalName());
                System.out.println("Distance between = " + distance);
                System.out.println("Current optimized  Val = " + optimizeVal);
               //  System.out.println("Sub Service Matchings = " + subServiceMatchings);
                System.out.println("Sum Unit Charge = " + sumUnitCharge);
                System.out.println("Avg Unit Charge = " + avgUnitCharge);
                System.out.println("                                                              ");

                //------------------------Optimization Logic Double.compare
                if (bestServiceProvider == null || Double.compare(optimizeVal, bestOptimizeVal)>=0) {
                  bestOptimizeVal = optimizeVal;
                  bestUnitCharge = avgUnitCharge;
                  bestServiceProvider = reply.getSender();
                  bestDistance = distance;
```

```
                bestRate = sRating;
                // This is the best offer at present
              }
          } catch (Exception e) {
              e.printStackTrace();
          }
        }
        repliesCnt++;
       /*if (repliesCnt >= serviceProviderAgents.length) {
            // We received all replies
            step = 2;
        }*/
      //  System.out.println("repliesCnt"+repliesCnt);

       if (repliesCnt >= serviceProviderAgents.length) {
            if (changeableCnt == changedCnt)
               step = 2;
            else {
                                step = 0;
                                repliesCnt = 0;
                                changedCnt++;
                                bestOptimizeVal=0;
                                System.out.println("Round: "+changedCnt);
                                //myAgent.setArguments(new Object[]{changedCnt});//----------------------XX

                                }     }

    } else {
        block();
    }
    break;
case 2:
    try {
        // Send the request order to the service provider that provided the best offer
        ACLMessage order = new ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
        order.addReceiver(bestServiceProvider);
        //order.setContent(targetService);
        order.setContentObject(serviceReqList);
        order.setConversationId("book-trade");
        order.setReplyWith("order" + System.currentTimeMillis());
        myAgent.send(order);
        // Prepare the template to get the service requests order reply
        mt = MessageTemplate.and(MessageTemplate.MatchConversationId("book-trade"),
              MessageTemplate.MatchInReplyTo(order.getReplyWith()));
        step = 3;
        break;
    } catch (Exception e) {
        e.printStackTrace();
    }
case 3:
    // Receive the service requests order reply
    reply = myAgent.receive(mt);
    if (reply != null) {
        // service requests order reply received
        if (reply.getPerformative() == ACLMessage.INFORM) {
            //S@6@VISHWA@192.168.8.100:1099/JADE
            sAgentInfo = reply.getSender().getName().split("@", 4);
```

```
                        try {
                            Connection myconnection = ConnectionManager.getConnection();
                            PreparedStatement queryU;
                            queryU = myconnection.prepareStatement("update distributedservicedb.userpool set OptimalServiceProvider= ?
,OptimalServiceProviderUserPoolID= ?,OptimalUnitCharge=? where UserPool_ID = ?");
                            queryU.setString(1, sAgentInfo[2]);
                            queryU.setString(2, sAgentInfo[1]);
                            queryU.setDouble(3, bestUnitCharge);
                            queryU.setString(4, vPoolId);
                            queryU.executeUpdate();

                            queryU = myconnection.prepareStatement("update distributedservicedb.userpool set status='X' where UserPool_ID = ?");
                            queryU.setString(1, sAgentInfo[1]);
                            queryU.executeUpdate();

                            myconnection.close();
                        } catch (Exception e) {
                            e.printStackTrace();
                        }

                        // service requests successful. We can terminate
                      if (targetService== null||targetService.isEmpty()){
                         System.out.println("Requester agent "+myAgent.getLocalName()+" ,"+ bestService + " is successfully consumed from
agent " + reply.getSender().getName());
                      }else{
                         System.out.println("Requester agent "+myAgent.getLocalName()+" ,"+ targetService + " is successfully consumed from
agent " + reply.getSender().getName());
                      }
                        System.out.println("                                                                    ");
                        System.out.println("-------------Optimum Values------------------'");
                        System.out.println("=====================================================================");
                        System.out.println("                                                                    ");
                        System.out.println("Best Service Provider = " + reply.getSender().getName());
                        System.out.println("Best Optimized Value = " + bestOptimizeVal);
                        System.out.println("Best Unit Charge  = " + bestUnitCharge);
                        System.out.println("Rate = " + bestRate);
                        //System.out.println("Location = " + bestLocation);
                        myAgent.doDelete();
                    } else {
                        System.out.println("Attempt failed: requested service already sold.");
                        System.out.println("                                                                    ");
                    }

                    step = 4;
                } else {
                    block();
                }
                break;
            }
        }
        public boolean done() {
            if (step == 2 && bestServiceProvider == null) {
                System.out.println("Attempt failed: " + targetService + " not available");
                System.out.println("                                                                    ");

                try {
                    Connection myconnection = ConnectionManager.getConnection();
                PreparedStatement queryU = myconnection.prepareStatement("update distributedservicedb.userpool set
OptimalServiceProvider= ? ,OptimalServiceProviderUserPoolID= ? where UserPool_ID = ?");
                queryU.setString(1, "NO");
                queryU.setString(2, "0");
                queryU.setString(3, vPoolId);
                queryU.executeUpdate();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return ((step == 2 && bestServiceProvider == null) || step == 4);
      }
   }  // End of inner class RequestPerformer
}
```

Appendix 02:

Message space spool output in Test case 01 in evaluation section

-----------------------Quoted from Message Space--

```
Found the following service provider agents:
======================================================================
S@52@Sanjaya@192.168.8.100:1099/JADE
S@53@Kapila@192.168.8.100:1099/JADE
Service Provider Agent = S@52@Sanjaya
Distance between = 4.766503373002384
Current optimized Val = 1.3228083033170898E-5
Sum Unit Charge = 1300.0
Avg Unit Charge = 650.0

Service Provider Agent = S@53@Kapila
Distance between = 4.833801968484621
Current optimized Val = 1.2683351574832852E-5
Sum Unit Charge = 1230.0
Avg Unit Charge = 615.0

Round: 1

Service Provider Agent = S@52@Sanjaya
Distance between = 4.766503373002384
Current optimized Val = 1.433042328593514E-5
Sum Unit Charge = 1200.0
Avg Unit Charge = 600.0

Service Provider Agent = S@53@Kapila
Distance between = 4.833801968484621
Current optimized Val = 1.2683351574832852E-5
Sum Unit Charge = 1230.0
Avg Unit Charge = 615.0
```

```
=====================================================================
Service Provider Agent S@52@Sanjaya@192.168.8.100:1099/JADE
terminating............................
Requester agent R@54@Saman, PLUMBER is successfully consumed from agent
S@52@Sanjaya@192.168.8.100:1099/JADE
-------------Optimum Values------------------'
Best Service Provider = S@52@Sanjaya@192.168.8.100:1099/JADE
Best Optimized Value = 1.433042328593514E-5
Best Unit Charge = 600.0
Rate = 2.5
Service Requester-agent R@54@Saman@192.168.8.100:1099/JADE terminating.....
```

--------------End---

Message space spool output in Test case 02 in evaluation section

---------------------------------Quoted from Message Space---

```
Hallo! Service requester-agent R@245@Meena@192.168.8.100:1099/JADE is ready.
Trying to serve PLUMBER
=====================================================================

Found the following service provider agents:
=====================================================================

S@244@Kalsha@192.168.8.100:1099/JADE
S@243@Nishantha@192.168.8.100:1099/JADE

Service Provider Agent = S@244@Kalsha
Distance between = 398.6363516047938
Current optimized  Val = 5.757332360618352E-8
Sum Unit Charge = 2000.0
Avg Unit Charge = 2000.0

Service Provider Agent = S@243@Nishantha
Distance between = 594.5601408095331
Current optimized  Val = 2.5734228378065712E-8
Sum Unit Charge = 3000.0
Avg Unit Charge = 3000.0

Round: 1
Current Unit Charge ::3000.0
Discount Amount::50
New unit Charge ::3000.0
Current Unit Charge ::2000.0
Discount Amount::60
New unit Charge ::2000.0
Current Unit Charge ::2000.0
Discount Amount::60
New unit Charge ::2000.0
Current Unit Charge ::3000.0
Discount Amount::50
New unit Charge ::3000.0
```

```
Service Provider Agent = S@244@Kalsha
Distance between = 398.6363516047938
Current optimized  Val = 5.757332360618352E-8
Sum Unit Charge = 2000.0
Avg Unit Charge = 2000.0


Service Provider Agent = S@243@Nishantha
Distance between = 594.5601408095331
Current optimized  Val = 2.5734228378065712E-8
Sum Unit Charge = 3000.0
Avg Unit Charge = 3000.0

=======================================================================

service PLUMBER is provided to agent R@245@Meena@192.168.8.100:1099/JADE
Service           Provider           Agent           S@244@Kalsha@192.168.8.100:1099/JADE
terminating................................

Requester    agent    R@245@Meena    ,PLUMBER    is    successfully    consumed    from    agent
S@244@Kalsha@192.168.8.100:1099/JADE

-------------Optimum Values------------------'
=======================================================================

Best Service Provider = S@244@Kalsha@192.168.8.100:1099/JADE
Best Optimized Value = 5.757332360618352E-8
Best Unit Charge  = 2000.0
Rate = 5.6
Service Requester-agent R@245@Meena@192.168.8.100:1099/JADE terminating.....
```
---End---

Message space spool output in Test case 03 in evaluation section

--------------Quoted from Message Space--

```
Hallo! Service requester-agent R@238@Silva@192.168.8.100:1099/JADE is ready.
Trying to serve
=======================================================================

Found the following service provider agents:
=======================================================================

S@236@Dilshan@192.168.8.100:1099/JADE
S@237@Mahinda@192.168.8.100:1099/JADE
Trying to serve
=======================================================================

Found the following service provider agents:
=======================================================================

S@236@Dilshan@192.168.8.100:1099/JADE
S@237@Mahinda@192.168.8.100:1099/JADE

Service Provider Agent = S@236@Dilshan
```

```
Distance between = 4.552717900851061E-4
Current optimized Val = 0.03734166608777492
Sum Unit Charge = 2700.0
Avg Unit Charge = 1350.0

Service Provider Agent = S@236@Dilshan
Distance between = 4.552717900851061E-4
Current optimized Val = 0.03734166608777492
Sum Unit Charge = 2700.0
Avg Unit Charge = 1350.0


Service Provider Agent = S@237@Mahinda
Distance between = 9.005919854101944E-4
Current optimized Val = 0.019987567257667822
Sum Unit Charge = 2550.0
Avg Unit Charge = 1275.0

Round: 1

Service Provider Agent = S@237@Mahinda
Distance between = 9.005919854101944E-4
Current optimized Val = 0.019987567257667822
Sum Unit Charge = 2550.0
Avg Unit Charge = 1275.0

Round: 1

Service Provider Agent = S@237@Mahinda
Distance between = 9.005919854101944E-4
Current optimized Val = 0.021415250633215525
Sum Unit Charge = 2380.0
Avg Unit Charge = 1190.0

Service Provider Agent = S@237@Mahinda
Distance between = 9.005919854101944E-4
Current optimized Val = 0.02216012891610998
Sum Unit Charge = 2300.0
Avg Unit Charge = 1150.0

Service Provider Agent = S@236@Dilshan
Distance between = 4.552717900851061E-4
Current optimized Val = 0.0389276055741283
Sum Unit Charge = 2590.0
Avg Unit Charge = 1295.0

Service Provider Agent = S@236@Dilshan
Distance between = 4.552717900851061E-4
Current optimized Val = 0.03969389702243791
Sum Unit Charge = 2540.0
Avg Unit Charge = 1270.0

Service Provider Agent S@236@Dilshan@192.168.8.100:1099/JADE
terminating................................

Requester agent R@238@Silva, CARPENTER is successfully consumed from agent
S@236@Dilshan@192.168.8.100:1099/JADE
```

```
-------------Optimum Values-----------------'
======================================================================
Best Service Provider = S@236@Dilshan@192.168.8.100:1099/JADE
Best Optimized Value = 0.0389276055741283
Best Unit Charge = 1295.0
Rate = 1.4
Service Requester-agent R@238@Silva@192.168.8.100:1099/JADE terminating...
```

Appendix 03:

Sample API URLs and Json request body

URL: http://localhost:8080/distributedServices/createUserPoolMerge

Request Body:

{"peopleId":1,
"agentContainerId":1131333,
"userFirstName":"SUNIL",
"loggingAs":"SERVICEPROVIDER",
"currentLocationLong":"-96.80322",
"currentLocationLan":"32.9697",
"mobility":"Y",
"trafficValue":131.0,
"timeMatter":"Y",
"knowService":"Y",
"userPoolStatus":"A",
"currentService":"PLUMBER",
"serviceChargeDiscount":100,
"description":"test",
"leakRepair":"Y",
"drainCleaning":"Y",
"toiletRepair":"Y",
"waterHeater":"N",
"gassLines":"N",
"doorWindor":"N",
"chairsTable":"N",
"tVRadio":"N",
"leakRepairUnitChg":1000,
"drainCleaningUnitChg":2000,
"toiletRepairUnitChg":300,
"waterHeaterUnitChg":null,
"gassLinesUnitChg":null,
"doorWindorUnitChg":null,
"chairsTableUnitChg":null,
"tVRadioUnitChg":null}