Republic of Iraq

Ministry of Higher Education and Scientific Research

University of Diyala

College of Sciences

Intelligent Agent Services in Distributed System

A Thesis

Submitted to the council of the College of Sciences- University of Diyala in a Partial Fulfillment of the Requirements for the Degree of Master in (Computer Science)

By

Reem Adil Qader Al-Zubidy

Supervised By

Professor

Naji M. Sahib

2020 A.D. *IRAQ* *1442 A.H.*

بسم الله الرحمن الرحيم

﴿نَرْفَعُ دَرَجَاتٍ مَنْ نَشَاءُ وَفَوْقَ كُلِّ ذِي عِلْمٍ عَلِيمٌ﴾

صدق الله العظيم

سورة يوسف

اية (76)

DEDICATION

To...

My family

My dear parents

My dear husband

My kid Yousif

All our distinguished teachers those who paved the way for our science and Knowledge.

Reem A. Qader

ACKNOWLEDGMENT

Initially and foremost I deeply thank God, Lord of the worlds, for all the grace and helping to bring this thesis to its end.

I would like to grateful my supervisor *Prof. Naji M. Sahib* for his support and encouragement. I am extremely thankful for his effort, completing and revising the thesis. My sincere thanks and gratitude is for the Council of the College of Science at Diyala University and the staff of Computer Science Department for all the supports to perform this study and for giving me the opportunity to have the M.Sc. level with great thanks to *Dr. Fatima M. Aboud* for all her supports and encouragement.

I would like to express my gratitude to my father and mother, my son, my brothers and their families and my sister who had unlimited support and patience.

There are not enough words to thank my dear husband *Kareem* for his logistic support and belief and encouragement throughout my studies. I pass many Thanks to my colleagues *Mustafa, Saja, Marwan and Meaad*.

Finally, I apologize for all those who not mentioned here and I thank them a lot.

Reem A. Qader

ABSTRACT

Global services with an agent or a multi-agent system are a promising and new research domain. However, several measures have been proposed to demonstrate the benefits of agent technology by supporting distributed services and applying smart agent technology in web dynamics. This thesis is a design to build a Semantic Web on the World Wide Web (WWW) to enhance the productivity of managing electronic library applications, where there is a problem that researchers and students endure, which is the process of exchanging books from e-libraries where they are slow or that the library needs large system data.

In this work a solution of this problem is found by using agent technology based on the “WebSocket,” any user can use this library to get fast communication and high information based on the model existing in the e-library, a simple and small model is considered. In addition, the library does not need an employee library responsible for entering information into the library database, as it was placed in Firebase it's a cloud-based database that synchronization data across every client in Real-time, and supply offline functionality. Any researcher can access the form by logging in. This application is installed on the central library server and every user who uses this library gets a quick result. This work is proven in our current thesis. In average 10 requests in HTTP take 25 ms and in Socket.io take 19 ms , while 100 requests in HTTP need 168 ms and in Socket.io take 30 ms, as well, 500 requests in HTTP take 779 ms and in Socket.io take 102 ms, in addition 1000 requests in HTTP take 1520 ms and in Socket.io need 172 ms, Therefore, The proposed model is

about 5-7 times faster than the model being used HTTP. Also, in this thesis, the data transfer process and loading benchmarks were calculated.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviations	Meaning
AI	Artificial Intelligence
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
BaaS	Backend as a service platform
BCT	Block Chain Technologies
CAI	Computer Aided Instruction
CBT	Computer Based Training
CCA	Client Checker Agent
CDN	Content Delivery Network
CERN	Conseil Européen pour la Recherche Nucléaire
CLI	Command Line Interface
CMD	Command Prompt
COSMO	Customer Services M. Owl
CPU	Central Processing Unit
CSS	Cascading Style Sheet
DAFFODIL	Distributed Agents for User-Friendly Access of Digital Libraries
DARPA	Defense Advanced Research Projects Agency
DLs	Digital Libraries
DRA	Database Repository Agent
ECMA	European Computer Manufacturers Association
eLib	Electronic Libraries
FTP	File Transfer Protocol
GCP	Google Cloud Platform
GEMs	Global education management systems
HTML5	Hypertext Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPs	Hyper Text Transfer Protocol Security
I/O	Input /Output
IA	Intelligent Agent
IAG	Intelligent Agents Group
IB	Intelligent Building
ID	Identification Data
IDS	intelligent distributed systems
IETF	Internet Engineering Task Force
IOS	iPhone Operating system
IoT	Internet Of Things
IT	Information Technology
ITS	Intelligent Transportation Systems

Abbreviations	Meaning
JISC	Joint Information Systems Committee
JIT	Just-in-time
JOSN	JavaScript Object Notation
JS	JavaScript
MAS	Multi-Agent Systems
MAST	Multi Agent Service Testing
MCA	Middleware Controller Agent
MUA	Mobile Urgent Agent
NLM	National Library of Medicine
NOSQL	Not Only SQL
NPM	Node Package Manager
PC	Personal Computer
PIN	Personal Identification Number
Pip	Pip Installs Packages , Pip Installs Python
PSSs	Power System Stabilizers
RDBMS	Relational DataBase Management System
RDF	Resource Description Framework
RFC	Request for comments
SDK	Software development kit
SMS	Short Message Service
SQL	Structured Query language
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UI	User Interface
URL	Uniform Resource Locator
VR	Virtual Reality
W3C	World Wide Web Consortium
WS	Web Socket
WWW	World Wide Web
XML	Extensible Markup Language

LIST ALGORITHMS

Chapter One

Introduction

Chapter One
Introduction

1.1 Overview:

Artificial Intelligence (**AI**) has revolutionized information technology (**IT**). **AI** is a subfield of computer science that includes the creation of intelligent machines and software that job and interact like human beings [1]. The field of **AI**, goes further still: **AI** attempts not just to understand but also into building smart entities. **AI** is one of the modernistic fields in engineering and science [2]. Work started in earnest soon after World War II, 1956 John McCarthy created the term Artificial Intelligence [3]. The subject of artificial intelligence (**AI**) is rooted in different research disciplines, such as philosophy [4, 5], computer science [6, 7], or futures studies [8, 9].

AI research is discrete into different research streams [10]. These streams vary on the one hand as to the topical of **AI** application (acting vs. thinking), on the other hand as to the type of decision making (an ideal, rational decision vs. targeting a human-like decision). This distinction leads to four research flows which are depicted in Table (1.1) According to the "Cognitive Modeling" (i.e. thought humanly) stream, an **AI** must be a machine with the mind [11]. This also contains performing human thinking [12], not only based on the same product as a human when given the same input but likewise on the same logic steps which led to the conclusion [13]. The "Laws of intellect" stream (i.e. thinking rationally) requires an **AI** to reach the rational decision despite what humans might answer.

Table 1.1: AI research streams based on Russell & Norvig [10].

Objective / Application to	*Humanly*	*Rationally*
Thanking	**Cognitive Modeling**	**laws of thought**
Acting	**Turing Test**	**Rational Agent**

Therefore, **AI** must follow-up the laws of thought through using computational models [14] which mirror logic. The "Turing Test" (i.e. acting humanly) includes that an **AI** must act intelligently when interactive with humans. **AI** must perform human tasks at minimum as pretty as humans [15]. These requirements can be tested by means of the Turing Test [16] finally, the "Rational Agent" flow considers **AI** as a rational [10] or intelligent [17]. This agent does not only act autonomously but also to achieve the rationally ideal score. **AI** and its Applications gets used in different fields of a lifetime of humans such as in Education, Computers have been used in education for over 20 years. Computer-based training (**CBT**) and computer-aided instruction (**CAI**) were the first such systems deployed as an attempt to teach using computers [18]. Application of artificial intelligence techniques in the design of power system-

stabilizers (**PSSs**), network penetration detection, in the medical field, in accounting databases, and in computer games, etc. [19].

In 1996, Broadcom Ireland formed a research collaboration with the Department of Computer Science at Trinity College Dublin, to explore the research in the domain of Intelligent Agents **(IA)** to use this new technology to applications in communications. The result of this collaboration was called the Intelligent Agents Group (**IAG**), Intelligent Agents are important topics in Information systems at the moment, and the first goal of **IAG** review is to report on researches in the accelerated evolution area of software agents and to highlight the possibility of application of this technology [20].

The agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors [21]. A human agent has ears, eyes, and other members for sensors, and legs, mouth hands, and other parts for effectors. A general agent is represented in Figure 1.1.

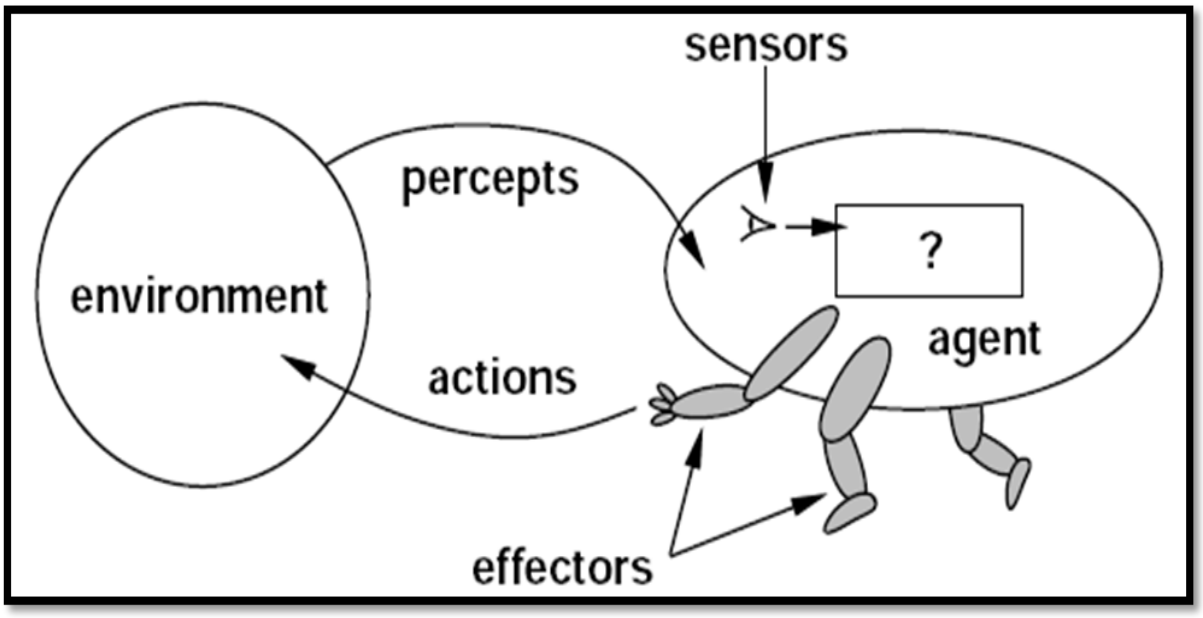


Figure 1.1 Agents interact with environments through sensors and effectors [21]

Applications of the intelligent agents can be classified according to their type of agent, by the domain of application and by technologies that used to implement the agent. The aim of producing this classification is simply to give a feel for the variety and breadth of agent applications. Here are the main applications of intelligent agents such as Industrial, Commercial Medical, Entertainment, and Comments Applications [22].

An intelligent agent is an autonomous computer system that has a smart technique that suite the environment to solve complicated problems. Any agent interacts with the environment through an agent life cycle, which means observe by sensing, deciding, and taking an action. To solve the most complicated problems that cannot be solved through a single smart entity, the modern approach agent services are developed. The distinguishing features of Agents are agents having an individual viewpoint and local control to solve a part of the problem meanwhile, all agents decentralized data and asynchronous processing in the agent's society. Smarter entities within a society can change the way of interaction between individuals belonging to this society. Human beings, intuitively, follow smarter people and adapt their behavior. An agent society is a society of software modules with socialization capabilities and two types of behaviors: firstly the individual behavior that launches from knowledge base built upon the creation of the entity plus reforming occurred due to the interaction with the environment. Secondly, social behavior is the dominant behavior [23].

Many software systems are generally distributed systems that are run on a broadly integrated group of the processors of a network [24, 25], many authors [26,27] have achieved research to test distributed systems that involve issues such as security, concurrency, timing and, controllability. Others [28, 29] have concentrated their

research on improving forward, the interpretation of the testing is prepared in order to make it adaptive and faster. The following results were obtained from the model In average 10 requests in HTTP take 25 ms and in Socket.io take 19 ms, while 100 requests in HTTP need 168 ms and in Socket.io take 30 ms, as well, 500 requests in HTTP take 779 ms and in Socket.io take 102 ms, in addition, 1000 requests in HTTP take 1520 ms and in Socket.io need 172 ms.

1.2 Related Works:

Several researchers have expressed interest in the intelligent agent, as it has an important role in all aspects of life, and here are some published works related to the current works:

- Sharples *et al.,* 1999 [30]: They discussed the learning in building govern systems, and contrast this process with existing intelligent building (**IB**) solutions. The authors clarified the importance of acquiring information from sensors to specific user needs. There is a new approach to the systems of **IB** that utilizes an intelligent agent process to autonomously controlling the building environment. They describe how their architecture, consisting of distributed agents and they show how these agents, employing a behavior-based process derivative from robotics research, are capable to learn and adapt to persons within a building. Finally, they show that such a system can be used to provides support for older people and allowing them greater quality and independence of life.

- El Yamany *et al.,* 2006 [31]: They proposed three –tire distributed system in software testing including server, middleware, and clients. The agent software represents a general agent and a mobile agent. The general agent applied in a framework that

consists of three-level agents. The first agent is a Database Repository Agent (**DRA**) within the server-side which response to monitoring data in the distributed application. Second, the middleware controller agent (**MCA**) in the middleware section which is the core stage of the framework proposed. **MCA** agent made the investigation process and create an integrated report by collecting the feedback from clients to conclude the process. The last agent located on the client-side. It's a client checker agent (**CCA**) and unit testing. Further testing is done by using Mobile Urgent Agent (**MUA**).

- Bai *et al.*, 2006 [32]: The authors developed an approach based on [25] to apply web service testing in a distributed system. The main idea is forced on classifying the test agent in different roles and the communication process is done by Extensible Markup Language (**XML**) based agent protocol.

  The Testing phase includes different tasks such as web testing creation, compact test planning, etc. in other words, every agent is responsible for one task. However, the tested agents are adaptive services taken by defining the tools which are used to communicate and generate each agent. While the rule-based system is proposed for test planning and agent communication. The three parallel and iterative stages are done in the testing process including test script. In summary, the Multi-Agent-based Service Testing (**MAST**) approach is concerned to support the universal testing and agent's classification process.

- Ma *et al.*, 2007 [33]: The researchers proposed an approach based on serving the communication between autonomous service agents and cross-enterprise applications by using a fabric agent. The fundamental autonomous system design is applying in the

fabric agent building process and is applied as message routing communication, highlighting the communication trust between agents in this work.

- Samanidou *et al.*, 2007 [34]: They gave a summary of the Donangelo- Sneppen model of the monetary exchange comparing it with related models in finance and economics literature. Their work dealing with many microscopic (agent-based) models of financial markets that have been studied by physicists and economists over the last decade. Their first aim is to reproduce thereby, and, provide explanations for the crashes and stunning bubbles seen in a historical episode, but they lack an estimation in terms of the global statistical lineaments of financial time series. This subject pursued by a number of contributions appearing in both the economics and physics literature since the late nineties decades. From the abundance of different multi-agent models that have manifested by now Levy, Solomon, Lux- Marchesi, and Huang models. Research questions are discussed in their concluding section.

- Liu, 2011[35]: The author aimed in his work to provide a comprehensive literature review on the use of intelligent agent technology in the library environment. The majority of the literature covers digital libraries (**DLs**) and there are fewer studies about services in traditional libraries. The application of agent technology in libraries is still at the experimentation and research stage. The survey has practical implications for libraries, librarians, and computer professionals in developing projects which employ intelligent agent technology to meet end-users' expectations as well as to improve information services within limited resources in library settings. This paper provides a comprehensive survey on the development and research of intelligent agents in libraries.

- Calvare *et al.*, 2018 [36]: The researchers present a systematic literature review of studies involving Multi-Agent Systems (**MAS**) and BlockChain Technologies (**BCT**) as solutions. The technology of MAS is widely used for the development of intelligent distributed systems (**IDS**) that manage sensitive data (e.g., healthcare, ambient assisted living, energy trading). Recent trends recommend to use **BCT** for **MAS**. Aim to provide a comprehensive overview of their application domains, they analyze assumptions, motivations, requirements, limitations and strengths presented in the current state. Moreover, discussing the future challenges, they introduce their vision on how **MAS** and **BCT** could be combined in different application scenarios.

- Khan and Bhatti, 2018 [37]: The aim of this work is to test the use of Semantic Web technologies for digital libraries. It also investigates the conceptions of university academicians and librarians in Pakistan about Semantic Web technologies and its use in digital libraries. Analysis of interview data was done to obtain results. The results of this paper showed that Semantic information, Dura Cloud, Onto Edit, and resource description framework (**RDF**) are the different Semantic Web applications that can be useful for digital libraries to develop semantic relationships among the digital contents and increase their accessibility in the web environment. The obtained results revealed that Semantic Web gave us precise results and meets the needs of user information in an effective way, also showed that next-generation of digital libraries use context-awareness technology, detecting sensors and software of intelligent agents to analyze user information needs and provide dynamic services. This paper conceives the future services of digital library and Semantic Web applications that can be used in a digital library.

- M¨uller *et al.*, 2019 [38]: The authors proposed an individual logic-based mechanism that amends reliability information to the data shared among the **MAS**. If multiple agents report the same event, their information is fused. In order to maintain high reliability, the machine detects and isolates misbehaving agents. Therefore, an attacker model is specified that includes faulty as well as malicious agents. The mechanism is applied to Intelligent Transportation Systems (**ITS**) and it is shown in a simulation that the approach scales well with the size of the **MAS** and that it is able to efficiently detected and isolated misbehaving agents.

- Bottino and Battegazzorre, 2019 [39]: The author focused to create the behavior of other non-human actors and it is a part of a larger project that worked on by multiple persons. The aim of the project was to make a virtual reality simulation to aid the therapists in studying and treating patients with apprehension disorders. The patient takes control of an avatar via a virtual reality (**VR**) headset and controllers and is free for moving in the store, communicating with other cashiers and customers, and also with different inanimate objects such as shopping carts or products on sale.
  The achieved results appear a good groundwork to improve upon. Especially due to the maximum modularity in both the behavioral aspect and the environment were of the most important in his project.

1.3 Problem Statement:

In electronic libraries, researchers and students are facing a problem in the process of exchanging books from electronic libraries as it is slow and the library also needs a large database to put electronic resources.

1.4 The aim of the thesis:

In this work, a small model was designed to be placed inside the electronic library in order to speed up the process of exchanging books from electronic libraries. Where were used agent technology depending on the WebSocket protocol, to obtain fast communication and high information, and any research and any user can access the model and it gets a quick result.

1.5 Motivation:

A recommendation is a web application that is so related to the consumer rather than the marketer, this is the challenge that is facing the recommended issue due to the wide spectrum of interests and demands of the cyber market. Different approaches and methodologies are deployed to cope with the rapid changes in market demands and interests. Expressiveness and interactions between the consumer and the recommendation are not the most important issue nowadays, the more important is the relation and the product, being advertising to the universe.

1.6 Thesis Organization:

The remaining parts of the present work include the following chapters:

Chapter Two: Theoretical Background

It contains an extensive overview to artificial intelligence, agent, and web socket technique. In addition, it presents the criteria utilize, background for the necessary techniques, and tools used in the present work.

Chapter Three: Design and Implementation of the Proposed Model

This chapter includes the steps of the proposed intelligent agent services system, describes the algorithms to perform that model.

Chapter Four: Experimental Results

This chapter shows the experimental work and the results which are acquired from the model running and the measures of the results of the test.

Chapter Five: Conclusions and Suggestions for Future Work

This chapter contains the conclusions which are obtained from the results of the present work and suggestions for future works.

Chapter Two

Theoretical Background

Chapter Two
Theoretical Background

2.1 Introduction:

In this chapter the basic theoretical aspects of artificially intelligence, and intelligent agents in distributed systems in libraries are given, a brief introduction to artificial intelligence, agent, and intelligent agents system are presented, the web socket technique, and traditional software vs software agent.

In addition, it presents the criteria used in intelligent agents, hypotheses to be investigated, systems in the libraries, the Agent Languages, background for the necessary techniques, and tools used in the present work.

2.2 Artificial Intelligence:

Artificial intelligence is a field of computer science and it is considered one of the most powerful and advanced science, in which many contemporary types of research are conducted. Research in the branches of artificial intelligence is still underway because the researchers see a brilliant future, it can determine what the world will be in the distant tomorrow. **AI** can be described in twain ways: (i) as a science that aims to discover the core of intelligence, and develop intelligent machines; or (ii) as the science of finding methods into solving complex problems that cannot be solved without applying intelligence [40]. Artificial intelligence has many branches, we mention them:

a) Machine Learning
b) Neural Networks

c) Robotics
d) Intelligent Agent
e) Expert System
f) Fuzzy Logic
g) Natural Language Processing

2.3 Agent:

An agent is a system of a computer that is situated in some encoding environment and capable of autonomous action in this environment in order to meet the objectives of its design. Autonomy used to express the truth that agents have the ability to act (to person implement actions) without the need for the intervention of humans or other systems.

The agent's necessity features are adaptive to changes in the environment and cooperation with other agents. Interacting agents associated in more complex societies – multi-agent systems. These groups of agents acquire several advantages in distributed systems, the flexibility of the software system engineering, mandate of sub problems on other agents [41].

2.4 Intelligent agent:

An intelligent agent (**IA**) is one that is able to flexible autonomous action to meet its design objectives, where flexibility means: reactivity (response to changes), pro-activeness (goal-directed behavior) and social ability (interaction with another agents). All these properties can be comprehended as requirements to an intelligent agent [41, 21].

I. Human-agent: has sensual organs such as ears, eyes, nose, skin, and tongue equivalent to the sensors, and other organs such as legs, hands, mouth for effectors.
II. Robotic agent: substitute cameras and infrared range finders for the sensors, and different engines and actuators for effectors.
III. Software agent: has encoded bit strings as its actions and programs.

2.4.1 The Structure of Intelligent Agents:

The structure of an agent can be viewed as [21]:

- Agent (Agent program + Architecture)
- Architecture: The machinery that an agent executes on.
- Agent Program: An implementation of an agent function.

2.4.2 The Classes of Agent:

There are five classes of intelligent agents based on their recognized intelligence and potential capability [42, 21, 2].

I. Simple reflex agents

Simple reflex agents act only on the basis of the current percept, ignoring the rest of the percept history. The agent function is based on the condition-action rule: "if condition, then action".

This agent function only succeeds when the environment is fully observable. Some reflex agents can also contain information on their current state which allows them to disregard conditions whose actuators are already triggered.

Infinite loops are often unavoidable for simple reflex agents operating in partially observable environments. Note: If the agent can randomize its actions, it may be possible to escape from infinite loops, as shown in Figure 2.1.

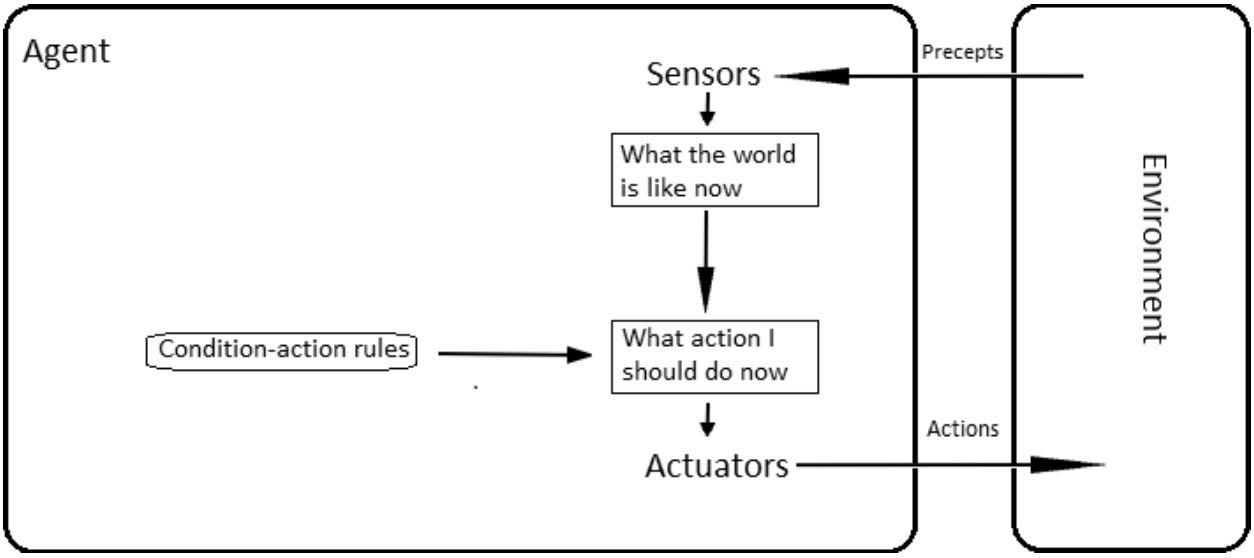


Figure2.1 Schematic diagram of a simple reflex agent [42].

II. Model based reflex agent

A model-based agent can handle partially observable environments. Its current status is stored inside an agent maintaining some type of structure that describes the part of the world which cannot be visible. This knowledge around "how the world works" is called the model of the world, hence the name model-based agent.

A model-based reflex agent must preserve some sort of interior model that relies on the percept history and reflects onto least some of the unobserved aspects of the current state. Percept history and the impact of the action on the environment can be

determined through using internal model. It then selects an action in the same way as a reflex agent. The agent may also utilize models to predict and describe the behaviors of other agents in the environment, as shown in Figure 2.2.

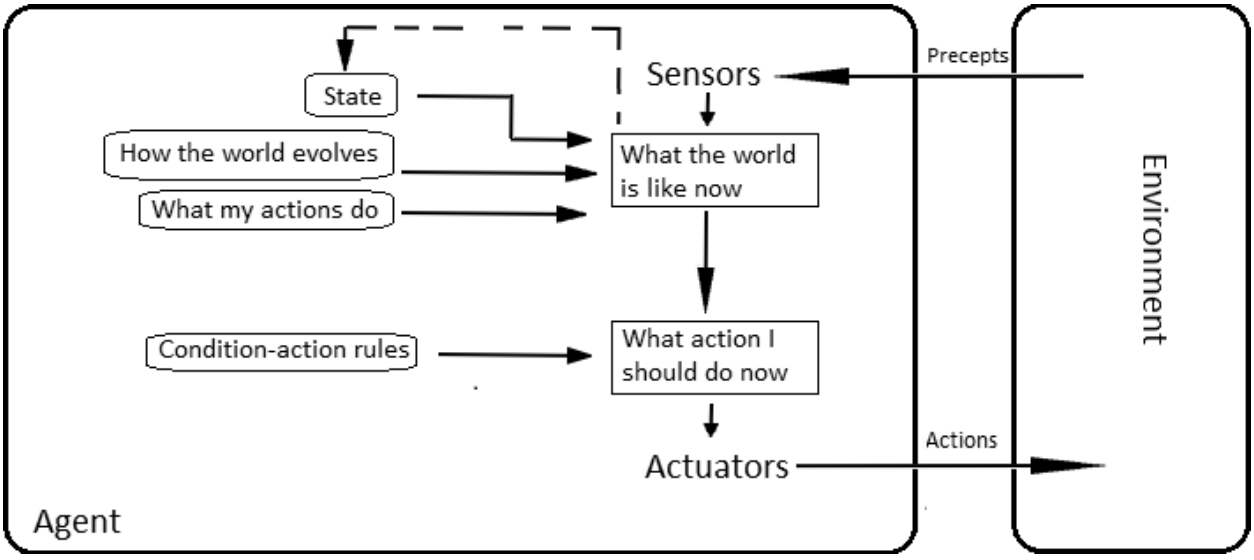


Figure2.2 A model-based reflex agent [21].

III. Goal-based agents

Goal-based agents further expand on the capabilities of the model-based agents, by utilizing goal information. Goal information describes cases that are desirable. This allows the agent a road to select among various possibilities, choosing the one which reaches the goal state. Planning and search are the subfields of **AI** devoted to finding work sequences that realize the agent's goals as shown in Figure2.3.

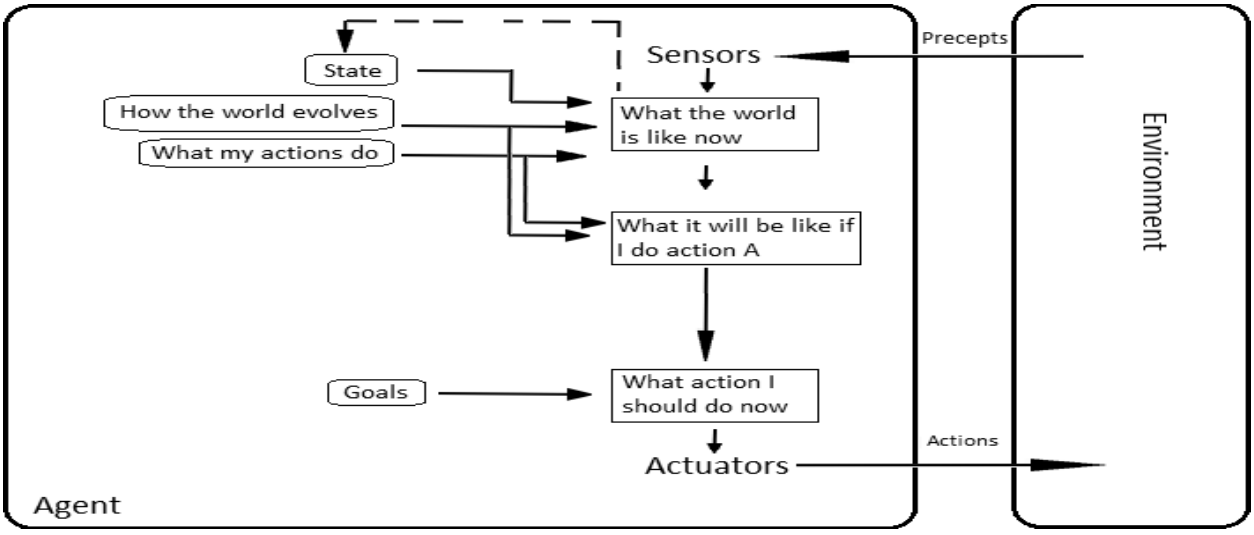


Figure 2.3 a model-based, goal-based agent [2].

IV. Utility-based agent

Goal-based agents only characterize between goal states and non-goal states. It is possible to locate the measure of how desirable a specific state. This measurement can be obtained during the use of a utility function which maps a state to a measure about the utility of the state. More general performance measurement should allow a comparison of various world states according to precisely how happy they would make the agent. The expression utility can be utilized to describe how "happy" the agent is.

The rationalistic utility-based agent selects the action that maximizes the predictable utility of the work results - that is, what the agent foresee to derive, on average, given the utilities and probabilities of each result. The utility-based agent has to model and keep track of its environment, tasks that have involved a big deal of research in perception, reasoning, representation, and learning as shown in Figure 2.4.

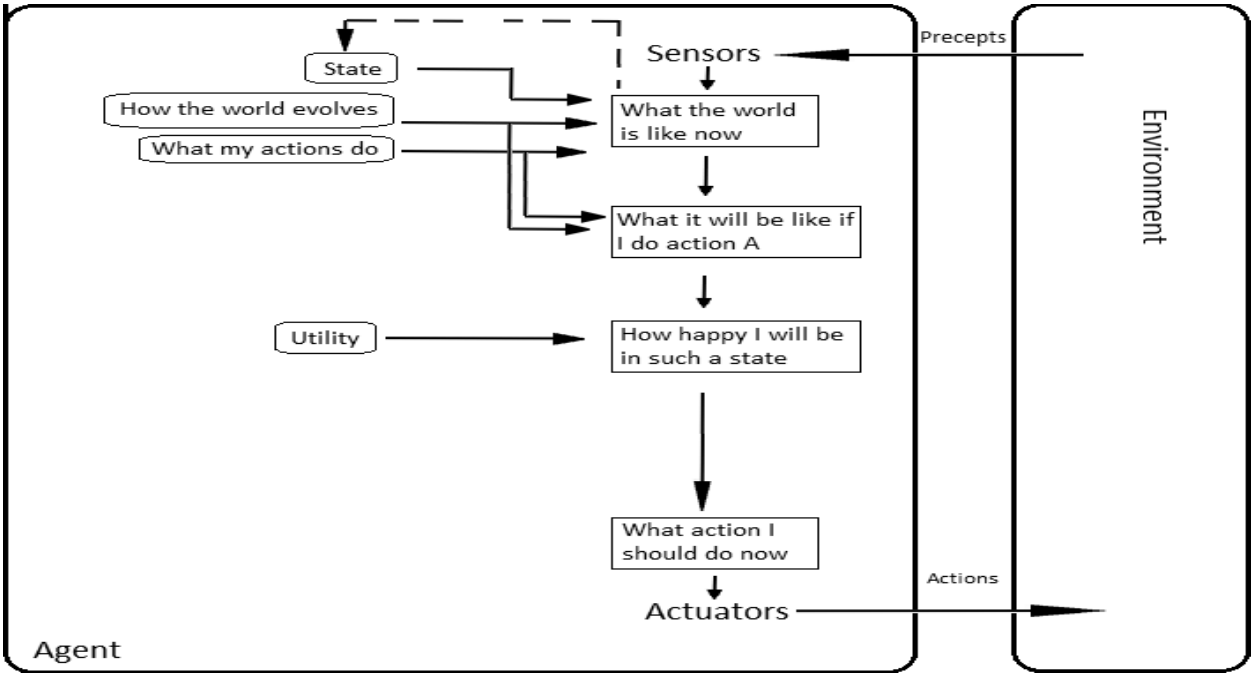


Figure 2.4 a model-based, utility-based agent [21].

V. Learning agents

Learning has the feature that it allows agents to initially run in unknown environments and to become more competent than its initial knowing alone might allow. The most important difference is among the "learning element", which is responsible for action improvements, and the "performance element", which is responsible for choosing outer actions.

The learning element utilizes feedback from the "critic" about how an agent is doing and determines how the performance element should be adjusted to do best in the future. The performance element is what we have previously rate being the entire agent: it takes in decides and percepts on actions. The last component of the learning

agent is the "problem generator". It's responsible for propose actions that will lead to novel and informative experiences as shown in Figure2.5.

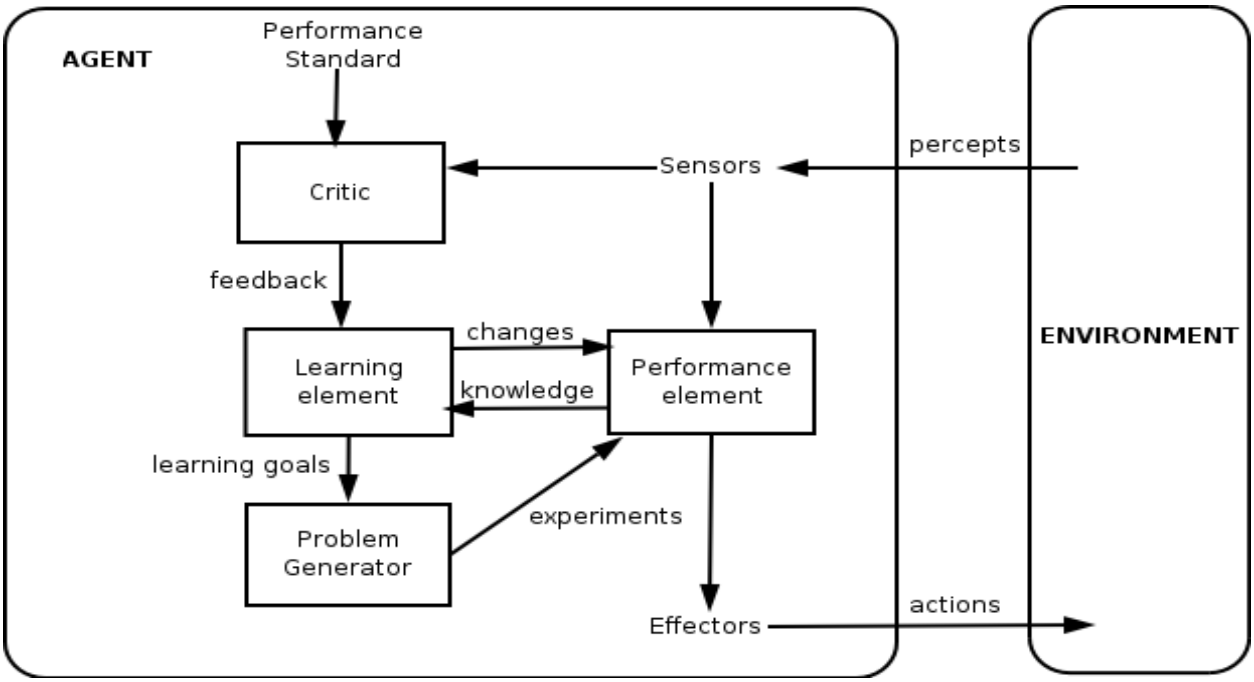


Figure2.5 A general learning agent [42].

2.5 The WebSocket:

The WebSocket protocol is an extension of Hyper Text Transport Protocol security **HTTP(S)** based standard that can full-duplex real-time communication between a web-server and its clients, typically uniform browser applications [43] as shown in Figure 2.6.

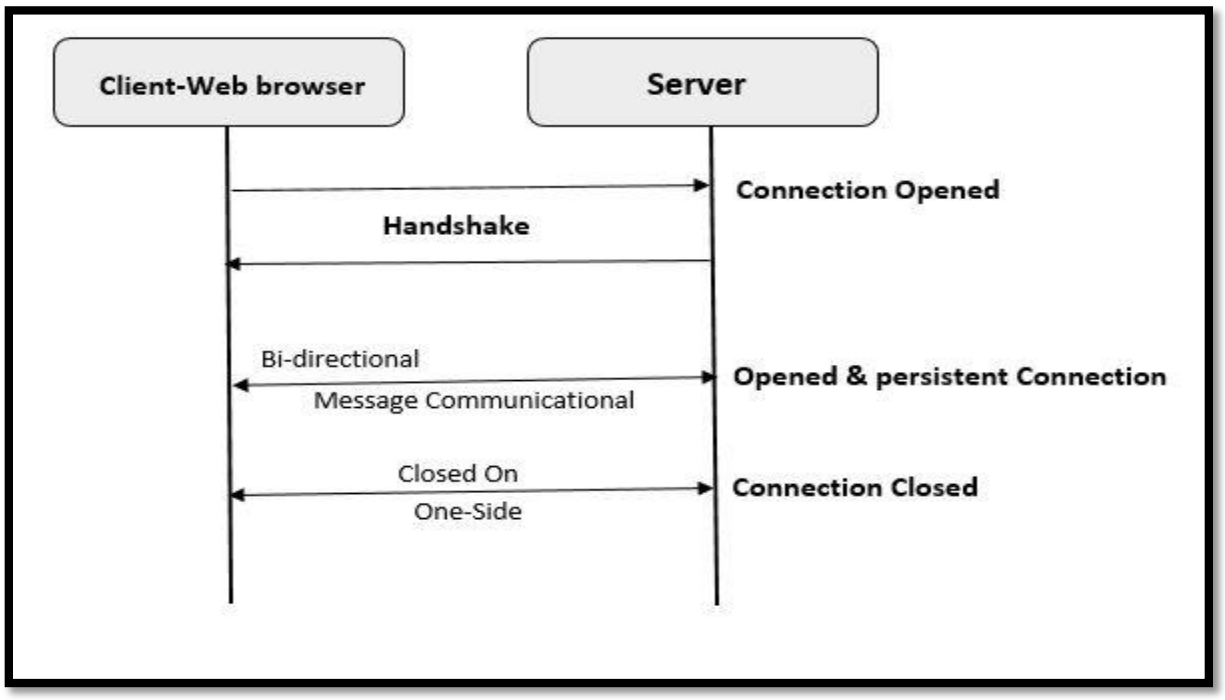


Figure 2.6: The WebSocket structure [43].

The Web-Socket Protocol is designed to replace present bidirectional communication technologies that utilize **HTTP** as a transport layer to interest from existing infrastructure (filtering, proxies, and authentication). These technologies were executed as trade-offs between reliability and efficiency because **HTTP** was not initially intended to be used for bi-directional communication. The Web-Socket Protocol tries to address the goals of present bi-directional **HTTP** technologies in the context of the present **HTTP** infrastructure; as such, it is designed to work through **HTTP** ports 80, and 443 as well as to backing **HTTP** [44].

WebSocket is a technology that provides a bi-directional real-time communications channel over a Transmission Control Protocol (**TCP**) connection [46,45].The WebSocket technology covers JavaScript **API** (Application

Programming Interface) that is specified by World Wide Web Consortium (**W3C**), and a protocol that is specified by Internet Engineering Task Force (**IETF**) [47,45]. WebSocket **API** is often denoted as **HTML5** WebSocket **API**, although it is actually separated from the **HTML5** specification and is currently being developed and specified independently [48,45].The WebSocket protocol is a network protocol running on **TCP**. It is designed to be implemented in web browsers and web servers but it can be used for other purposes as well. The WebSocket protocol is independent of Hyper Text Transfer Protocol (**HTTP**), although the protocols have similarities such as using **TCP** port 80 and the handshake process when initializing a connection [49,45].

2.5.1 Features and advantages of WebSocket:

The main difference in WebSocket – compared to the usual network traffic over **HTTP** – is that the WebSocket protocol does not follow the traditional request-response convention [50]. Once a client and a server have opened a Web-Socket connection, both endpoints may asynchronously send data to each other. The connection remains open and active as long as either the client or the server closes the connection. [50] In contrast, the traditional approach relies on polling. That is, a client opens a new **TCP** connection and makes an **HTTP** a request to receive data from a server. This requires several round-trips between the client and the server before any actual information is sent. Moreover, a new request is needed by the client for every separate data transmission. Hence, compared to WebSocket, the traditional **HTTP** request-response convention results in high latency and high amount of network traffic [51]. Figure 2.7 illustrates how the WebSocket communication differs from traditional **HTTP** calls.

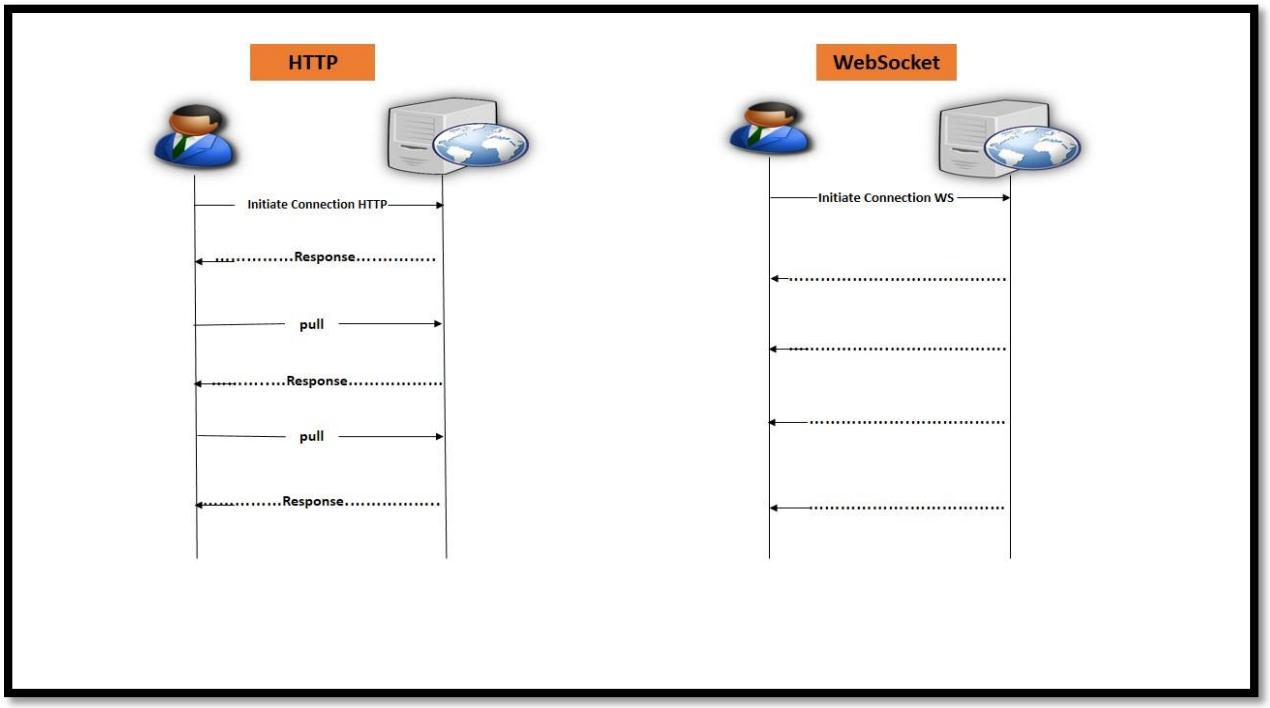


Figure 2.7: HTTP vs WebSocket [45]

2.6 Web Servers:

The use of the Internet in our day-to-day duties is becoming very common. Nowadays, the usage of the Internet is increasing as more people have the opportunity to access it. And, this is thanks to the deployment of many different services, such as online shopping, the e-libraries, information consultant, or media-related.

The Internet was developed by **DARPA** (Department of Defense of the United States of America) in the ’60s. But, its purpose was to be a military network where exchange confidential information. It did not exploit to be a communication tool since the end of 1990 when Tim Berners-Lee (while he was working for the **CERN**) begun the development of the first browser, called “Worldwide Web”, which has been the seed for further browsers and web development. The Internet has evolved to become

a very important communication way, displacing in some cases more traditional ways of communication such as readable, radio, or television [52].

Programming for the web is like programming for the computer environment, as the latest features introduced on the web are very similar to those found on the desktop. There are different programming languages used on the web (e.g., PHP, Java, Python, etc.), and all of them belong to high-level programming languages. **HTML** language, which is used in all the web pages, is a markup language [53] and therefore, is not considered as a programming language. Moreover, accessing the content on the Internet can be done by different means, such as **HTTP** (Hypertext Transfer Protocol), **FTP** (File Transfer Protocol), **SSH** (Secure Shell), etc. Although, the most used in order to surf the web is **HTTP**.

2.6.1 Web content classification:

A. Static content

Static web content consists of showing a web page as equally as it is stored in the webserver. It does not offer any user interaction delivering the requested file without any change. Later on, when the web browser receives the file it displays it in a user-friendly way. Some examples of this static content classification can be **HTML** only web pages, as well as images.

This type of content is equal for all the users viewing the web page, no offering personalized web navigation. But, on the other hand, it is not computational expensive for the webserver, as it does not need to process anything.

B. Dynamic content

Dynamic web content is those web pages which offer user interaction or are generated by the webserver. In this case, the webserver must offer some kind of language interpreter or compiler to process the requested web page file.

Moreover, it can be configured to display different content to different users, allowing a more personalized web experience.

There is two different dynamic content web pages classification, regarding the dynamic content which is being used. The first one, are those web pages using client-side scripting and content creation. In this case, dynamic behavior occurs within the presentation layer in the client's web browser, in response to some actions, such as mouse over a section, keyboard actions, or timings. Some examples of client-side dynamic languages are JavaScript or Action Script.

Usually, this dynamic language is embedded into the **HTML** web page, and it is being processed by the web browser once it is shown.

Meanwhile, server-scripting and content creation are those programs being executed on the web server side and used to personalize the user's navigation experience. The response web page offered by the webserver is triggered by some **HTML** form post parameters, **URL** parameters, or simply by the type of web browser. Those dynamic content web pages are used with server-side languages such as PHP, Python, Java, etc. [52].

2.6.2 The Advantages of the webserver:

- All files are stored in a central location
- Network peripherals are controlled centrally
- Backups and network security is controlled centrally
- Users can access shared data which is centrally controlled

2.7 The difference between a Web server (HTTP) and WebSocket (WS):

The difference between the proposed model based on WebSocket and the previous techniques based on HTTP shown as Table (2.1).

Table 2.1: Explains the difference between the WebSocket, and HTTP.

WebSocket	*HTTP*
WebSocket is bi-directional.	HTTP is a unidirectional communicational protocol.
WebSocket can be used in that case.	Http should not be used, when you don't want the connection to be opened for a long time.
WebSocket can be used with Chat, Trading, frequent updates kind of application, where a request is made very frequently.	Http works poorly with frequent kinds of applications, which overloads the server.
WebSocket, it keeps the connection open until the state has died.	Whenever a request is made through HTTP, it creates a connection at the client (browser) and closes it once the response from the server is received.
WebSocket uses ws protocol (like ws://www.google.com).	Http uses HTTP or https protocol for sending a request (like http://www.google.com).

2.8 World Wide Web:

World Wide Web is one of the services of the internet. It is a way of accessing integrated information in the form of web pages over the medium of the internet with the help of web browsers. According to [54] World Wide Web is a global network of internet servers that provides access to interlinked documents locally and remotely. It is a vast network of linked hypertext files stored on computers throughout the world that can provide computer users' with information on a huge variety of subjects. The information can be in the form of regular text, hypertext, pictures, sounds, use net newsgroups, and other types of data. To access such information from web use client program is necessary like Internet Explorer, Firefox, etc. The web uses **HTTP** protocol language over the internet to transmit data. On the web, each web page can hold not only information but also links to other pages. In each page, a particular word or group of words are highlighted and there is a link between the highlighted item and other information, this is called hypertext[55].

2.9 Traditional software vs software agent:

The main difference between traditional system and agent system as shown in Table (2.2) [56]:

Table 2.2: Difference between traditional system and agent system [56].

Regular software	Agent software
Static	Dynamic
Direct	Indirect

Regular software	Agent software
Never change – human operator or program error	Adapts , Learning
Inflexible	Flexible
Run on time	Continue to run over time
Stay on same place	Travel to other server
Access data by network	Can interact and communicate with local entity

2.10 The criteria used in the intelligent agent:

While evaluating a particular software project the scope of software and the end-user are the most important considerations. Software architecture, integration and documentation support are a few other important considerations.

Where it will be used a WebSocket for fast communication and used Node.js to be fast access and easy access.

2.11 Hypothesis to be investigated

Along with the implementation of the proposal, the following hypothesis is to be investigated to reveal crucial standards in applying agent services in a distributed system.

a. Knowledge developed faster in a society with an applied intelligent agent system.

b. A distributed system that possesses an intelligent agent's behaviors produces a fast and accurate conclusion with decreasing the error rate.

2.12 The systems in the libraries:

The concept of agent technology has practical implementation within computing science and, engineering, but it also has big potential to affect the way work is done in libraries, as the aim of improving the research experiment and search outcome for users. Professional agents can negotiate and manage the transmission of information, and this ability makes it beneficial for task computer-based, library-related functions, and tasks, as well as its support ending user.

The Nationalistic Library of Medicine utilizes **COSMO** to provide an automated reply to users' questions [57, 58]. Librarians at the **NLM** continually survey question and answer register and supply new scripts for **COSMO**. The agent allows users to obtain more particular answers to questions when searching the Internet to detect too many results, and also allows users to request sensitive questions that they might not otherwise ask the librarian [59, 58].

An intelligent agents might be utilized in a library setting, including mediation between the user and information, automated serials processing, virtual reference,

and automated interlibrary loaning processing, circulation, acquisitions, and digital libraries. In addition, patron information administration, cataloging, and online interacting tutorials are also a domain where agents might be useful and reduce workload [60, 58]. And suggested that the agents of information gathering could

contribute to the base of knowledge to be used. The uses of agent technology are not widespread in libraries. Using agent technology to work in libraries means cooperation with systems staff, part of librarians, and to some extent the user.

There are numbering of examples where agent architecture has been applied successfully in library settings, include the United Kingdom-based on Joint Information Systems Committee (**JISC**), Electronic Libraries Program (**eLib**) project **MALIBU**, the University of Michigan Digital Libraries (**UMDL**) initiative, and the **DAFFODIL** digital library system. The **UMDL** remains one of the maximum initial digital library projects.

2.13 Agent Languages:

The requirements of this language are not accurately defined. At least, the language must provide a structure that corresponds to the purpose, and capability of the agent [41]. As agent technology be most established, we might foresee to view a variety of software tools become obtainable for the construction and design of agent-based systems. The requirement for software backing tools in such an area was identified as long since as the mid-1980s [62, 61]. The up growth of a number of prototypical agent languages is one mark that agent technology is becoming most great used and that numerous more agent-based applications are probable to be developed in the near futurity. “By an agent language, we mean the system that authorizes one to program software or Hardware computer systems regards some of the concepts developed through agent theorists” [63, 61]. At the very least, we expect this a language to guaranty several structures corresponding to the agent. However, we

might too expect to see some other attributes of agency (goals, beliefs, or another mentalistic concept) utilized to program agents.

2.14 Background, necessary techniques and, tools:

The web is a global hypertext system, has evolved into a distributed (distributed client-server) application platform, in which the application logic and user interface are two separate entities. Web applications are a critical part of internet infrastructure where huge applications now days are using this platform to accomplish its business logic over the globe. The tendency toward web applications is rapidly increasing and networking sessions are abstracted as web services called by web application clients to accomplish tasks. The web services are accessible anywhere in the world; with this opportunity resources for computing could be unlimited due to huge amounts of services distributed over the internet. Cloud computing is a marketing term for technologies that provide computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services[23].

The service suppliers are publishing their services in the service guide and service consumers must query this guide for what services are obtainable [64].

We used WebSocket technique to build an agent, and we also used many of the most important tools:

i. JavaScript will be implemented in this work

JavaScript, a supported language scripting by all modern browsers, websites can collect information about the obvious implementation of the browser. Furthermore, JavaScript permits us to gain details concerning the host system, e.g., the operating system installed plugins and the screen resolution, and this could be used to fit a website to the properties of a user's environment and device, providing an optimal user experience [65].

- **The difference between JavaScript and Java**

JavaScript must make us believe that JavaScript is related to the Java language. Both of them are C-based and used in the web of client-side applications, in the following paragraph it'll explain some differences between them:

It cannot deny the fact that both Java and JavaScript are languages of programming used to develop different features or applications of web pages, except for the codes that are different for each of them [67].

The main difference is the type of application used the run of Java applications is either through the webpage or from the desktop, which is considered as stand-alone programs that usually opens in an autonomous program window. If Java didn't install, it won't be able to run its own applications on a device While JavaScript is included in all modern browsers so the user will be able to use it readily when loading a web page. Using JavaScript, the user can build different applications on the same web page, and he can also develop work and video games on them in the window of the browser.

While Java can use to develop huge programs, it usually requires a large memory that makes the computer run slowly or slow down another program. Moreover, JavaScript works with small memory, so it is in many webpages because of the ability and low memory requirements to supply many unique features [68, 66].

ii. Firebase database

The Firebase Real-time Database is a **NoSQL** cloud-based database that asynchronization data across every client in Real-time, and supply offline functionality. Data is stocked in the Real-time database as **JSON**, and all linked clients share one instance, automatically receiving updates with the newest data [69]. The difference between Firebase, **MySQL**, and **SQL**ite database shown in Table (2.3) [70].

Table 2.3: The difference between Firebase, MySQL, and SQLite [70].

Firebase	*MySQL*	*SQLite*
Firebase it's a cloud service Also Firebase is a NoSQL database. In Firebase Data stored / processed in a cloud.	MySQL is a relational database management system (RDBMS).	SQLite is local database on Android device (data stored/processed on a device) with SQL interface
Firebase is suitable for real time applications.	MySQL mostly used for relational data and transactions.	Today several browser, operating system and embedded systems use SQLite it's a most deployed database engine.
Firebase is owned by Google	MySQL It's open source	SQLite is in the public domain Its a free for use for any purpose
Firebase is only available on GCP (Google Cloud Platform)	MySQL install it anywhere and several cloud providers support managed version of it	SQLite is file-based the database consists of a single file on the disk, which makes it extremely portable and reliable.

➢ Features of Firebase

The features of Firebase are as below:

1. Real-time Database: It synchronizes the sender´s data to Firebase which updates instantly, and the client receives the updated data on real-time in any connected device.
2. File storage: Firebase enables the file storage directly from the client even in offline mode. It is implemented in Chat App as below:
   Data in the cloud is highly protected by Firebase´s own security system.
3. Authentication: Firebase provides its own pre-built or custom **UI** authentication or sign-in method either email / password, phone number, google **id**, etc.
4. Hosting service: Firebase offers the implementation of static websites and single-page applications which is built with **HTML**, **CSS**, and JavaScript. It uses HTTPS and SSL protocols for the security of files and data delivery. It does not require the Content Delivery Network (**CDN**), which is a built-in function in Firebase.
5. Free of cost: Firebase does not require payment for providing database service to client to some extent. On the other hand, the client has to pay if the database memory is not enough for the specific services. [71]

➢ Cloud Storage

The usage of Cloud Storage means that a user/customer/company will save their data within the cloud instead of on a local system. Access to the data is consummated by network connectivity and client service.

One advantage of Cloud Storage is that customers can access their data from any location, even if they do not have access to their organization's network [72].

- **Advantages of Cloud Storage**

There are many benefits to storing data in the cloud over local storage.

1. Companies only pay for the storage they use.
2. The data is quickly accessible and reliable. The data is located on the web across multiple storage systems instead of a local site.
3. Better protection in case of a disaster. Sometimes, the organization has a local backup and in cases of fire or natural disaster, the backup will not be available.
4. Cloud vendors provide hardware redundancy and automatic storage failover. This helps to avoid service outages caused by hardware failure.
5. Virtually limitless storage capacities. If the customer does not have the necessity of extra storage, the costs will decrease.
6. Workload balance. Cloud vendors help customers to achieve the best performance by balancing workloads.
7. A unified view of storage. Cloud vendors provide an export to get a unified view of storage utilization. [72] [73].

The architecture of cloud computing is defined by layers. There are three main layers: the cloud infrastructure (IaaS), cloud application platform (PaaS), and cloud application software (SaaS) as shown in Figure 2.8.

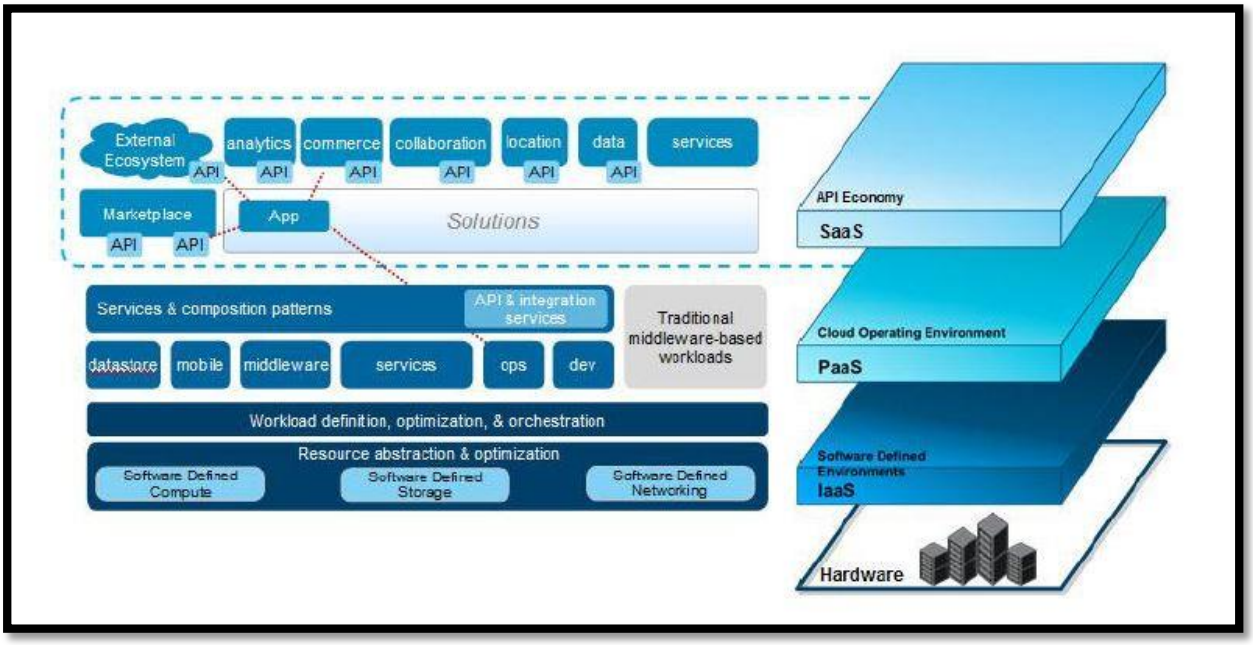


Figure 2.8:The Architecture of Cloud Computing [74].

The layer that includes cloud storage is Infrastructure as a Service (IaaS). Physical and virtual resources used to construct the cloud are included in this layer. Resources are provided and managed in physical and virtual servers [75].

➢ Package Manager:

It is a specific system or programming that manages all other external software in the software project. We do not have to search for Content Delivery Network (**CDN**) for a specific program every time or download it and then manually insert it and so on. It is sufficient that the programming is supported by the package manager Which we are going to work on, then download it by that package manager, after that it'll be automatically added to the project, moreover, its update can also be controlled, as the package manager helps to update all software with just one code

and adjust the compatibility of software in your project. also, it helps you to delete unused software in the project in order to organize the largest and most professional for the project software.

The package manager manages all the external resources in the project from adding external resources, frames, and frameworks to the open-source libraries that can be found on Github to include in the project [76].

All of these procedures can be done with one console line (on all platforms). The package manager has different types and variations as well as the field of their use:

1- **Composer:** The composer supports all types of Php-based programs or frameworks that work directly with it on a Php Native project or framework such as Laravel, Codeigniter, Symfony, and others. It adds all the files to a specific folder and calls them via Autoloader, which saves the trouble of calling each one separately.

2- **Gems:** the package manager dedicated to the Ruby programming language, and the Rails framework automatically performs the Gem.

3- **Npm**: It is a package manager for Node.js, which is based on JavaScript and can be used in almost any web project. It suffices to install node.js on the computer, then starts to work on it and gives us the ability to manage all the resources and packages in the program. Actually any project on web-based on JavaScript does not work without using Npm as the package manager.

4- **Pip:** since we are talking about package manager, it must be mentioned that software based on Python like Flash, Django, and others has its own package manager which is the pip. It allows us to install and manage any external packages as in previous packages.

iii. WebSocket Agent

The WebSocket **API** and convention are characterized in **RFC 6455**. WebSocket gives you a bidirectional, full-duplex correspondences channel that works over **HTTP** through a solitary attachment.

Existing hacks that run over **HTTP** (like long surveying) send demands at interims, whether or not messages are accessible, with no information on the condition of the server or client. The WebSocket **API**, nonetheless, is extraordinary the server and client have an open association from which they can send messages. For the security-disapproved, WebSocket additionally works over Transport Layer Security (**TLS**) or Secure Sockets Layer (**SSL**) and is the favored strategy [77].

iv. Node.js

Node.js is server-side scripting dependent on Google's **V8** JavaScript motor. It is utilized to assemble adaptable projects, particularly web applications that are computationally basic yet. You can utilize Node.js in creating **I/O** escalated web applications like video gushing destinations. You can likewise utilize it for growing: Real-time web applications, Network applications, General-reason applications, and Distributed frameworks. Node.js is quick and dependable for overwhelming records and substantial system load applications **IoT** [78].

Chapter Three

The Architecture of Proposed Model

Chapter Three
The Proposed Model

3.1 Introduction:

This chapter manages execution necessities, structure contemplations, and the means taken to make login or make sign up. It consists of describing the proposed model planning in addition to the implementation details for each stage related to the proposed model. The proposed model creates a fast electronic library using an agent-based on WebSocket technology. It's can be implemented using the central university library and linking all university colleges with it or implementing it in the library of the College of Science and linking the six departments of the college with it as explained in this chapter.

3.2 The architecture of the proposed model:

The proposed model consists of five parts (Users, WWW, Proxy Server, WebSocket, and Agent) as shown in Figure 3.1.

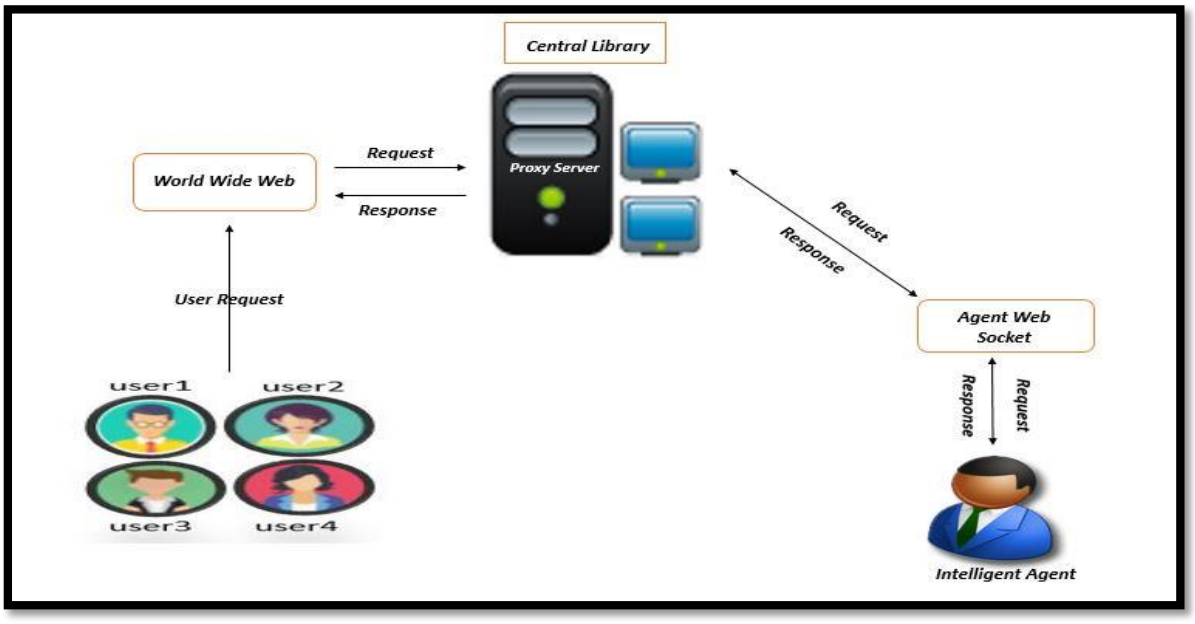


Figure 3.1 : Proposed Model Architecture.

3.2.1 Users or Clients

A user or client is somebody who utilizes a computer or network service. In general, users or clients of computer systems and software products lack the technical expertise needed to fully understand how they work.

A user or client often has a user account and is identified to the model by a user name (or surname). Other terms of username include the login name, the user name (or surname), account name, and handle.

Where access to the electronic library for the purpose of ordering the book.

3.2.2 World Wide Web (WWW)

The World Wide Web (**WWW**), commonly known as the Web, is an information system where reports and other web resources are distinguished by Uniform Resource Locators (**URLs**, such as https://www.example.com/), which may be interlinked by hypertext, and are accessible over the Internet.

The **URL** for the model is entered, and then we perform the registration process using the user's email and password.

3.2.3 Proxy server

A proxy is a server that acts as an intermediary of requests between a client that requests resources from other servers. It is simply another computer that acts as the hub through which requests are processed. When you connect to one of these servers, your computer sends your requests to the proxy server, which then processes these requests and then returns you to what you requested. In this way, it acts as an intermediary between your personal computer (**PC**) and the rest of the computers on the Internet.

3.2.4 WebSocket

WebSocket is a communications protocol for a persistent, bi-directional, **TCP** connection from a user's web browser to a server.

A WebSocket connection is initiated by sending a WebSocket handshake request from a browser's **HTTP** connection to a server to upgrade the connection. Along with the upgrade request header, the handshake request includes a 64-bit Sec-WebSocket-Key header. The server responds with a hash of the key in a Sec-WebSocket -Auth header. This header exchange prevents a caching proxy from resending previous WebSocket exchanges.

From that point, the connection is binary and does not conform to the **HTTP** protocol. A server application is aware of all WebSocket connections and can communicate with each one individually. When WebSocket is open, either the user or the server can send messages at all times until one of them terminate the session communication. On the other hand, standard HTTP allows users only to demand new data.

WebSocket server with express is **WS** in node.js server we can initiate the **npm** and install from **npm** express **WS**, in this situation when the WebSocket install as a server then we will have the client to install reconnecting-WebSocket **html5**-WebSocket this way the client will be available to access the WebSocket server.

This is the server:

```
var express = require('express');
var socket = require('socket.io');
// App setup
var app = express();
var server = app.listen(4000, function(){
    console.log('listening for requests on port 4000,');
});
```

For the client, we have added this in the web page for the client to access the WebSocket server.

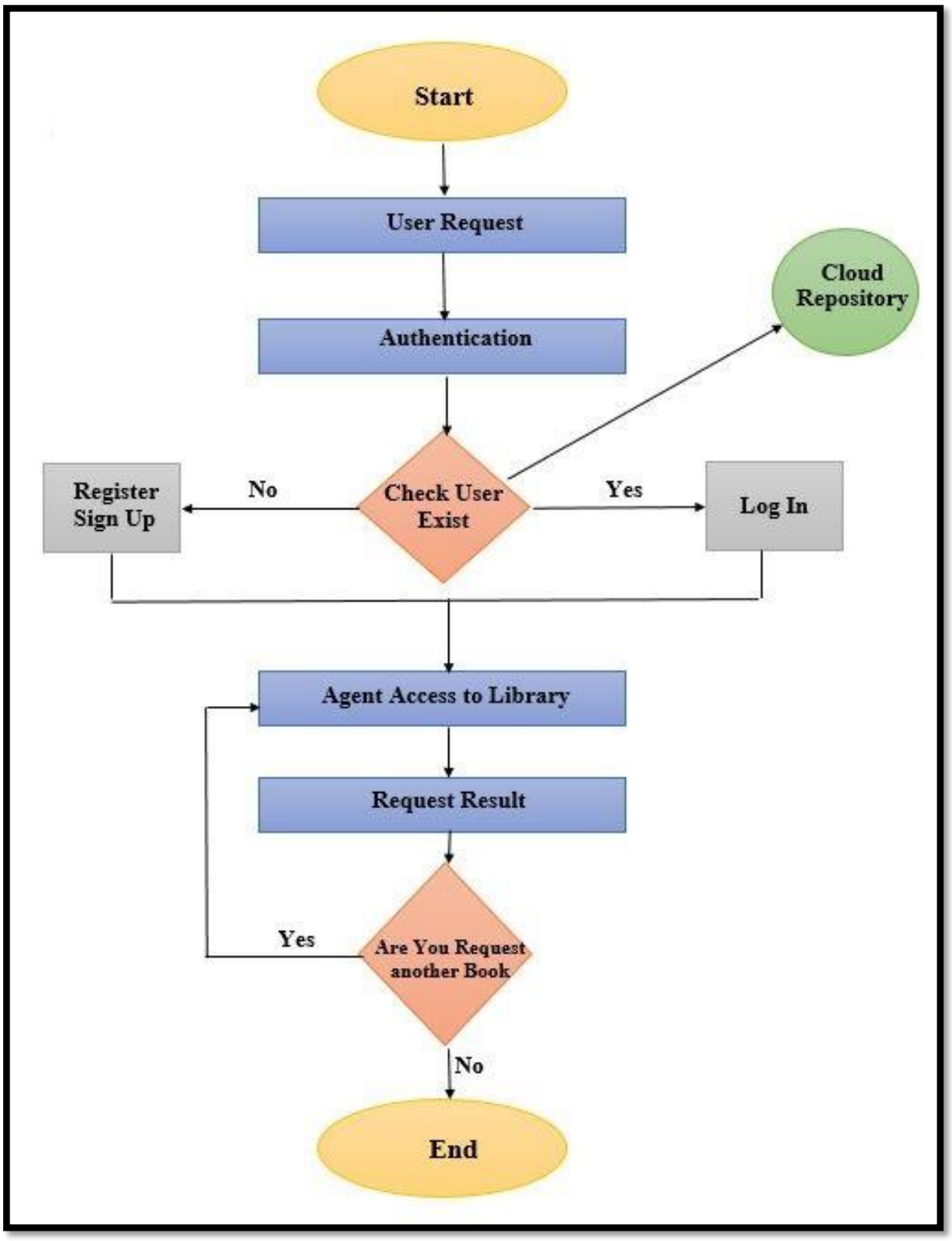
```
<scriptsrc="https://cdnjs.cloudflare.com/ajax/libs/socket.io/1.7.3/socket.io.js">
</script>
```

3.2.5 An agent

It is a computer system that is able to autonomous action in the environment to meet the objectives of the design.

3.3 The proposed model:

It includes making a fast model for a library and every user who uses this library takes the fast result, and below the flowchart of the proposed model as shown in Figure 3.2.



Finger 3.2: General Proposed model flowchart

3.3.1 Authentication:

Authentication is the testing of claimed user identity. It is used to establish a user's identity and make sure that the users are the same as those who log in. Authentication is the second step, after identification. There are many methods to perform authentication. For example, we can use passwords, tokens, or even biometrics. Authentication can take place locally on the system where we log in or transmitted over a network to a centralized authentication server. When we perform authentication, there are three different types that can be used.

The first type is something we know, like a **PIN** or a password. The second type is something we have, like a smart card, token, etc. The third type is something we are physical, like a fingerprint (biometrics) in this proposed model, the first type is used to demonstrate user credibility.

- **Passwords**

The most common form of type one authentication is a password. The password is simply a code or random string that we memorize. Passphrases are longer strings and are typically converted to a virtual password before sending it to the authentication server for validation. There is also something called cognitive passwords, which actually multiple questions are given to the user that only that users should know the answers to. Next, we have composition passwords, which are created automatically by a system. Also, there are one-time passwords, which are intended to be used only one time. Passwords should be strong enough to prevent easy guessing and easy cracking, but on the other hand, easy to remember so users won't forget them down.

3.3.2 Check User Exist:

- **Login**

Login simply means to validate the user like authenticating the user. It means the user is identified and authenticated to access the website or a program where he/she is already registered. Login works both in web portals and web applications.

By logging in what you simply did is authenticating yourself to gain access to your library account. It is just an action that identifies you as a returning user rather than sees you as a new user, in which case you have to sign up.

- **Sign up**

Sign up is an action to register yourself for a new account. Different web portals might use different terms for returning users but they all use "sign up" for the process of first-time registration.

You need to sign up before you can access your library account which requires you to fill your details like email **ID**, and a password to log in. If you are not a registered user already, you are always required to register.

The main difference between the two lies in the term itself. You log in to officially record your presence as a returning user, whereas sign up simply registers your intent to be present officially in the system. Sign up means to create an account as a new user so that you can log in later with your credentials.

Sign out:

When a user wishes to move out of the website, they have to use the sign-out.

3.3.3 Agent access to the library:

When entering the library after the authentication process. We should define the category whether it is buying, borrowing, or renting a book. Then we enter the personal email address of the client in the username field, after that, the name of the required book should be written in the messages field, or respond to another client requesting a book from another user. Then we click the submit button to send the request.

3.3.4 Request result:

Each library will have an e-mail system, the library or the branch that owns the requested book sends the e-book by e-mail to the authorized of the client who wants to buy, borrow or rent.

The library is sending the link of the book with scheduling (i.e. a specific date for renting or borrowing the book) Example, from 1/ 1/ 2020 - 10/ 1/ 2020 in which the borrower or the client who rented the book has the right to use it in the specified period, after that date, the book will not be available and cannot be accessed.

In other words, cookies become expired. Since the term cookies is a file when we enter the browser (the library agent) it will give us a file to download to the browser. Cookies will create a relationship between the user of the browser and the library, meaning that they have become known in the model because of the file that was loaded on the browser by cookies.

3.4 JavaScript:

JavaScript was at first made to "make web pages live". It tends to be composed legitimately into the **HTML** of a website page and automatically began when the page is loaded. Scripts are given and executed as plain text, as it developed, JavaScript turned into a completely independent language with its own detail called **ECMA** Script.

Today, JavaScript can implement not only in the browser, yet in addition to the server, or in fact on any device that has a private program called the JavaScript engine. For this reason, the JavaScript language was used to implement this model, as it was used on the client and server-side.

3.4.1 Advantages of using JavaScript in-browser:

Present-day JavaScript is a "safe" programming language. It doesn't give low-level access to memory or **CPU** since it was at first made for browsers that don't require it. JavaScript's capabilities greatly depend on the environment it's running in. For instance, Node.js supports functions that allow JavaScript to perform network requests, read/write arbitrary files, etc.

In-browser JavaScript can do everything identified with communication with the client, web page control, and the webserver.

For example, in-browser JavaScript can add new **HTML** to the page, modify styles, and, change the current substance. Pointer movements, run on mouse clicks, key presses, react to user actions. Send off requests over the network to distance servers, download and upload files (so-called **COMET** and **AJAX** technologies). Obtain and set cookies, ask questions to the guest, display messages.

3.5 Firebase platform and the main goal:

Firebase was founded in 2011 by two developers named James Tamplin and Andrew Lee, and three years later it was acquired and developed by Google in order to give developers a ready and scalable (Backend) infrastructure to support their own projects of whatever size.

This type of platform is known as Backend as a service platform (**BaaS**) dealing with this platform is easy and does not involve any difficulty, because, through Firebase **SDK**, developers can communicate with the Firebase programming interfaces from the client-side, and for each target platform (Web, Android, **IOS**) there is its own **SDK**.

Thanks to the Firebase platform, the web application developer (and mobile as well) can be a comprehensive developer able to accomplish a complex and integrated application from scratch to production without worrying about the server issue and the associated infrastructure.

3.5.1 The firebase services:

There are many services that Firebase provides. Which was mentioned in the previous chapter, in this chapter we will mention an important service for Firebase, namely Firebase remote configuration :A cloud service that enables developers to control a number of settings for their applications and configure them without forcing users to update those applications on their devices.

Hosting Firebase is not exclusive to anyone and everyone can use it. All we need is an account in the Firebase platform that can be obtained easily.

Then we need to prepare the private site that you want to upload in hosting, and finally, we follow the following steps:

Install tools firebase Command Line Interface (**CLI**) it is the command line receiver that we use a lot in dealing with many projects on the web and others.

For example, in building projects, we control with Node.js, which is the most popular.

Before installing tools Firebase **CLI**, we will first need to install Node.js.

After completing the node.js installation process, we execute the following command:

Npm install-g

Firebase-tools

By doing this, the Firebase **CLI** tools will be installed in the computer through the package manager Npm provided by the node.js that we have installed previously.

3.5.2 Node Package Managers (NPM)

Npm stands for Node Package Managers. It is the world´s largest open-source library which is used for Node.js It was developed by Rebecca Turner, Kat Marchan in 2010. It is written in JavaScript itself and free for use. It is highly recommended to download Nodejs directly from its official website i.e. http://nodejs.org. NPM comes with the installation of Node.js.

3.6 Node.js:

Beginning of 2010, JavaScript achieved a significant and significant leap, especially after Google developed the fast V8 engine, which Ryan Dal built on the Node.js platform that gave JavaScript another dimension where we can use JavaScript outside browsers and deal with servers directly, thus developing fast and efficient applications using only one language which is JavaScript.

The high speed of Node.js is mainly due to two factors: the advanced Chrome V8 engine and the mechanism **I/O**, which is called in English non-blocking versus the blocking mechanism adopted by other languages.

3.6.1 JavaScript V8 Engine:

Node.js uses the V8 JavaScript engine developed by Google Chrome programmers. It is characterized by great strength and speed that enables it to implement JavaScript codes faster and lighter thanks to its dependence on advanced technology in the compilation process called **JIT** Compilation (or Just In Time Compilation) that converts the program into codes that the machine understands during the implementation of the program and not before starting implementation it was unlike before the V8 when most engines adopted the traditional Interpretation mechanism.

3.6.2 Blocking and Non-Blocking Mechanism

JavaScript is a programming language based on Events. This is why Node.js has developed a non-blocking concept platform.

To illustrate this, we will see two examples illustrating the difference between the two mechanisms: Obstructive Mechanism Vs Obstructed Mechanism

In the **PHP** language, for example, that adopts the blocking model, when starting the process of raising two files, the program must wait for the end of uploading the first file in order to start uploading the second file, and this is like blocking the program as it remains stuck on a certain point until it is completed to pass for another task.

Whereas in Node.js the matter is completely different, the latter depends on the non-blocking mechanism where the program can start the process of raising the two files at the same time and at the end of the upload process we resort to the callback functions for each file to do what we want as shown figure 3.3.

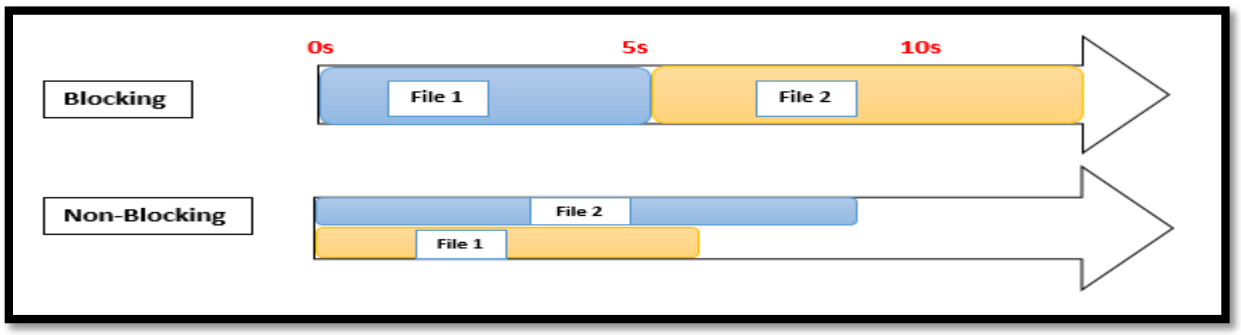


Figure 3.3: The Blocking and Non-Blocking

Node.js saves us from wasting time and allows us to do other things while waiting for the time-consuming tasks to end.

It is true that working on Node.js is more complicated compared to programming languages that compete with it, such as **PHP**, but it remains a revolutionary and promising solution for applications that require special speed and real-time interaction between the user and the server.

3.7 Atom 64 bits:

Atom 64bit is a text editor that's new, a tool you can customize to work anything but also utilize productively without ever touching a configuration file. You select from thousands of open source bundles that add new functionality and features to the application or build a bundle from scratch and publish it for everyone else to utilize. The software comes pre-installed with four **UI** and eight syntax themes in both light and dark colors. Atom Free Download last Version for Windows **PC**. It is a full offline setup installer of the tool.

It's easy to style and customize Atom 64bit. You can't weak the feel and look of your user interface with **CSS**/Less and add main features with **HTML** and JavaScript. Atom is a desktop application built with **HTML**, JavaScript, **CSS**, and Node.js integration. It runs on Electron, a framework for building cross-platform apps using web technologies, figure 3.4 shown as Atom editor.

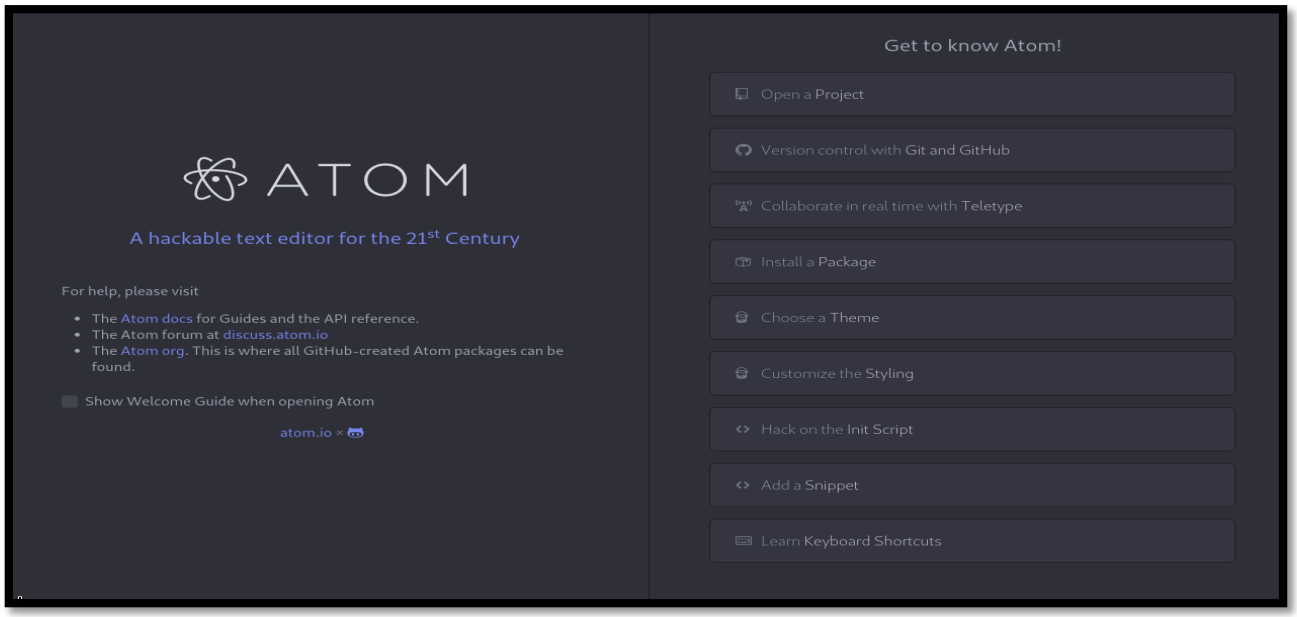


Figure 3.4 Atom Editor

3.8 The Socket.io library for building simultaneous web applications:

Socket.io is one of the most popular libraries in the Node.js environment, as it has made it easy for web developers to build asynchronous applications between server and client, especially relying on the WebSocket protocol. When we say real-time or simultaneous applications, we mean every application in which clients need to obtain

3.8.1 Working mechanism WebSocket:

In the central library, for example, users receive notifications of new comments and watch them in real-time on their screen without interference from them. What happens is that once we enter the library site, the latter opens a page that does not close until after exiting from it, and through it, the server can send certain data to the client who is eavesdropping on the server and waiting for any new news from him to display the data he receives on the user interface.

This process (Bi-directional), meaning that the server is also able to eavesdrop on the client and wait for new information from him to send it to other clients. This happens, for example, in chat or chat applications, where you write your message and directly when you press the send button it appears at the other end of your friend.

3.8.2 Socket.io library relationship to WebSocket:

The socket.io library allows us to use WebSocket technologies easily by providing a clear and simple programming interface, and it uses other methods and technologies to ensure complete compatibility with old browsers that do not support WebSocket. For example, if the browser does not support WebSocket and Flash is installed simultaneously, socket.io uses Adobe Flash Socket to make the connection.

In other cases, various techniques such as **Ajax** Long Polling or Forever Iframe are used.

All of these technologies are used by the library to target the largest possible number of devices and browsers, and we as researchers do not have to understand it because, in the end, we are dealing with the unified socket.io interface.

Also with this library it is also targeting the server-side (Node.js) along with browsers, so we will not need any other library to manage server-side operations. Here it should be noted that the server-side socket.io library can only be used with the socket.io library itself the client-side.

3.9 The proposed model structure in the library:

The obtained results could be applied at university libraries and connect them together, for example, the central library of Diyala University and the libraries in each college at Diyala University, as shown in Figure 3.5.

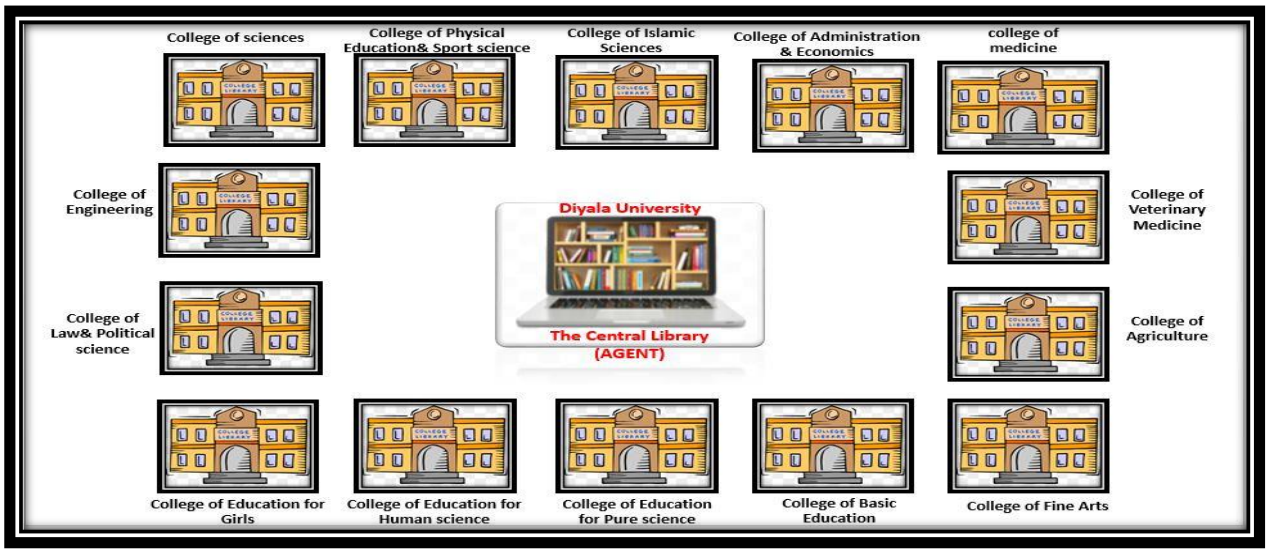


Figure 3.5: The structure of the Diyala University Central Library with college libraries

Or, the central library of the College of Science can be linked with the departments of the College of Science at Diyala University as shown in Figure 3.6.

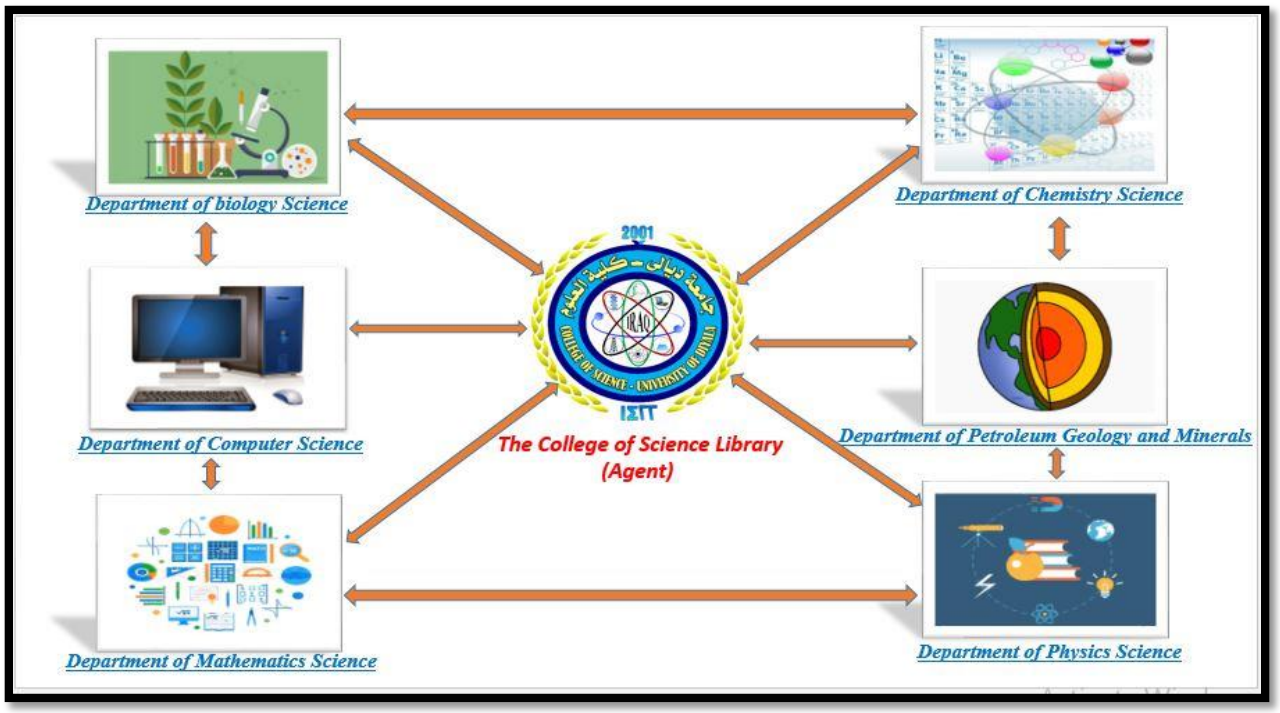


Figure 3.6: The structure of College of Science Library with its departments.

3.10. Firebase configuration:

After opening Atom editor and write in JavaScript language, it should be write the code for the username and password that is by configuration a firebase. During this work, Firebase used as a knowledge base and it is existing in the cloud.

```
var firebaseConfig = {
    apiKey: "AIzaSyDaepU5bdcVL_B1JEHeXvgRqfQWPqhExEQ",
    authDomain: "access-agent.firebaseapp.com",
    databaseURL: "https://access-agent.firebaseio.com",
```

```
    projectId: "access-agent",
   storageBucket: "access-agent.appspot.com",
    messagingSenderId: "728655332862",
    appId: "1:728655332862:web:1d11d56b6a8434b35eacaf",
  measurementId: "G-52GDSXXTLQ"
};
```

3.10.1 Initialize Firebase:

Pseudo codes (3.1) : Initialize Firebase
Inputs: information of firebase Configuration **Output: Initialize firebase**

Begin
Step1: const txtEmail = document.getElementById('txtEmail');
Step2: const txtPassword = document.getElementById('txtPassword');
Step3: const btnLogin = document.getElementById('btnLogin');
const btnSignUp = document.getElementById('btnSignUp');
const btnLogout = document.getElementById('btnLogout');
 	
End

3.10.2 Login authentication:

Pseudo codes (3.2): Login code

Inputs: Enter email and password

Output: login to library

Begin

Step1: const email = txtEmail.value;
Step2: const password = txtPassword.value;
Step3: firebase.auth().signInWithEmailAndPassword(email, password)
Step3.1: .then(function(firebaseUser) {
window.location.replace("http://localhost:4000");
})

.catch(function(error) {
var errorCode = error.code;
var errorMessage = error.message;
window.alert("Message "+errorMessage);
Step4: window.alert("Message "+errorMessage);

End

3.10.3 Signup authentication:

Pseudo codes (3.3): Signup code

Inputs: Enter email and password

Output: Signup to library

Begin

Step1: const email = txtEmail.value;
Step2: const password = txtPassword.value;
Step3: firebase.auth().createUserWithEmailAndPassword(email,
password)
Step3.1: .then(function(firebaseUser) {
window.location.replace("http://localhost:4000");
})
.catch(function(error) {
var errorCode = error.code;
var errorMessage = error.message;
Step4 : window.alert("Message "+errorMessage);

End

3.10.4 Logout authentication:

Pseudo codes (3.4): Logout code
Inputs: Session of current user **Output: logout from web socket**

Begin
 	
Step1: firebase.auth().signOut();
Step2: firebase.auth().onAuthStateChanged(firebaseUser => {
if(firebaseUser){
Step3: console.log(firebaseUser);
btnLogout.classList.remove('hide');
 	
} else{
console.log('not logged in')
Step4 : btnLogout.classList.add('hide');
 	
End

3.11 Make a connection:

At this stage, the two-way communication is established between the page and the library.

3.11.1 Socket IO library:

Socket IO is a function located inside the WebSocket, using this function, it works with the handshake of the page with the WebSocket that means they become bi-directional.

```
var socket = io.connect('http://localhost:4000');
```

Here the category is set, message, and handle selection within the value selected, and all of them are part of the request appearing for each user.

Pseudo codes (3.5): Communication between users
Inputs: the input is the URL **Output: Library**

Begin
 	
Step1: socket.emit('chat', {
Step2: message: message.value,
Step3: handle: handle.value,
Step4 : category: valueSelected
});
 	
End

The Socket found in the index.js library and the socket library was used. This library was built with JavaScript language and it has many functions, such as

connection and communication. If I could not but the following codes, the web socket will not work.

```
var server = app.listen(4000, function(){
console.log ('listening for requests on port 4000,');
});
```

These are all found in Cloudflare when Cloudflare is downloaded it fixed at Socket IO and then it can access.

Pseudo codes (3.6):Establish connection
Inputs: Server **Output: User communication**

Begin
 	
Step1: var io = socket(server);
Step2: io.on('connection', (socket) => {
Step3: console.log('made socket connection', socket.id);
 	
End

3.11.2 Static and public files:

Inline 11 in index.js library we wrote:

```
app.use(express.static('public'));
```

The static the web socket, it means that the application is static and could not change, in another meaning, the information inside the application is static for every user, and whereas the public means that the information is public for all users.

An Express installation of the system was done. It is a layer or class that contains Express functionality to build my web application. The Socket IO will make Bi-direction communication.

Npm instal express

Npm instal socket io

3.12 Package. JSON

It is the logbook on what used in the application. The index.js was used as the basis, require express, socket.io, and socket security were used also. We built socket.io to exist and can request it in this application.

Pseudo codes (3.7): Package. Json
Inputs: Required packages **Output: Dependencies of the project**
Begin **Step1:** "express": "^4.17.1", **Step2:** "firebase-tools": "^7.11.0", **Step3:** "socket.io": "^2.3.0", **Step4:** "soket.io": "^0.0.1-security" **End**

3.12.1 Nodemon:

When the application runs the nodemon was used, when we wrote this index library, it reads from the nodemon to make available communication on the browser to open in the agent library. The nodemon work is available on the webserver for any client who wants to make access to the application. Since I use JavaScript when the client wants to log in to the model he will write (npm run serve), it is the nodemon who runs the service and this is the login to the application.

Pseudo codes (3.8): Nodemon
Inputs: Index.js **Output: Web application**

Begin
 	
Step1: {
"nodemon": "^1.19.4"
},
Step2: "scripts": {
"serve": "nodemon server.js"
},
 	
End

3.13 Agent Auth.html:

In the library, this code was used to establish connection and authentication to make them available to users.

Pseudo codes (3.9): Agent auth.html
Inputs: JavaScript SDK **Output: Authentication web page**
Begin **Step1:** <script src="http://www.gststic.com/firebasejs/7.6.1/firebase-app.js"></script> **Step2:** <script src="http://www.gststic.com/firebasejs/7.6.1/firebase-auth.js"></script> **Step3:** <script src="/www.gststic.com/firebasejs/7.6.1/firebase-firebase.js"></script> **End**

The core Firebase **JS SDK** is always required and must be listed first-->

Add **SDK**s for Firebase products that you want to use

http://firebase.google.com /doce/web/setup #available-libraries.

<scriptsrc="http://www.gststic.com/firebasejs/7.6.1/firebase-analytics.js"></script>

In this code, we establish a connection to the cloud of the Firebase.

Pseudo codes (3.10): Authentication settings
Inputs: User authentication data **Output: Enter the library**

```
Begin

Step1: <div class="container" align="center" margin="auto">

Step2:  <input id="txtEmail" type="email" placeholder="Email">
Step3: <input id="txtPassword" type="password" placeholder="password">

Step4: <button id="btnLogin" type="button" class="btn btn-action">Log
                                    in</button>
Step5: <button id="btnSignUp" type="button" class="btn btn-
secondory">Sign Up</button>

Step6: <button id="btnLogout" type="button" class="btn btn-
action">Log out</button>
</div>
Step7: <script src="app.js"></script>

End
```

In this code, we identify the authentication for the user to establish a connection with the cloud base on the email and password. Also, we made an alignment and center alignment and we made the margin automatic.

Chapter Four

The Experimental of Results

Chapter Four
The Experimental Results

4.1 Introduction:

In this chapter, been summarized the implementation results obtained by the proposed model described in detail in chapter three. The experimental results of the model phases will be explained in addition to a comparison process with other existing systems. This chapter will contain a detailed description of the steps involved in model implementation.

The proposed model is implemented in JavaScript programming, using a laptop. The experiments were performed on an Intel (R) Core (TM) i7-8565U CPU @ 1.80 GHz 1.99 GHz, 64 bit Operating System, and 8 GB RAM. In the following sections, the detailed steps and implementation results will be explained for each step to accomplish the suggested model.

4.2 Implementation requirements:

JavaScript programming, Firebase database, WebSocket Agent, and Node.Js are used with a laptop. It is very fast as it is measured in milliseconds according to the server size. Implementation results for each step will be explained to complete the proposed form.

Where we make copies of the path of the WebSocket playlist below as shown in Figure 4.1.

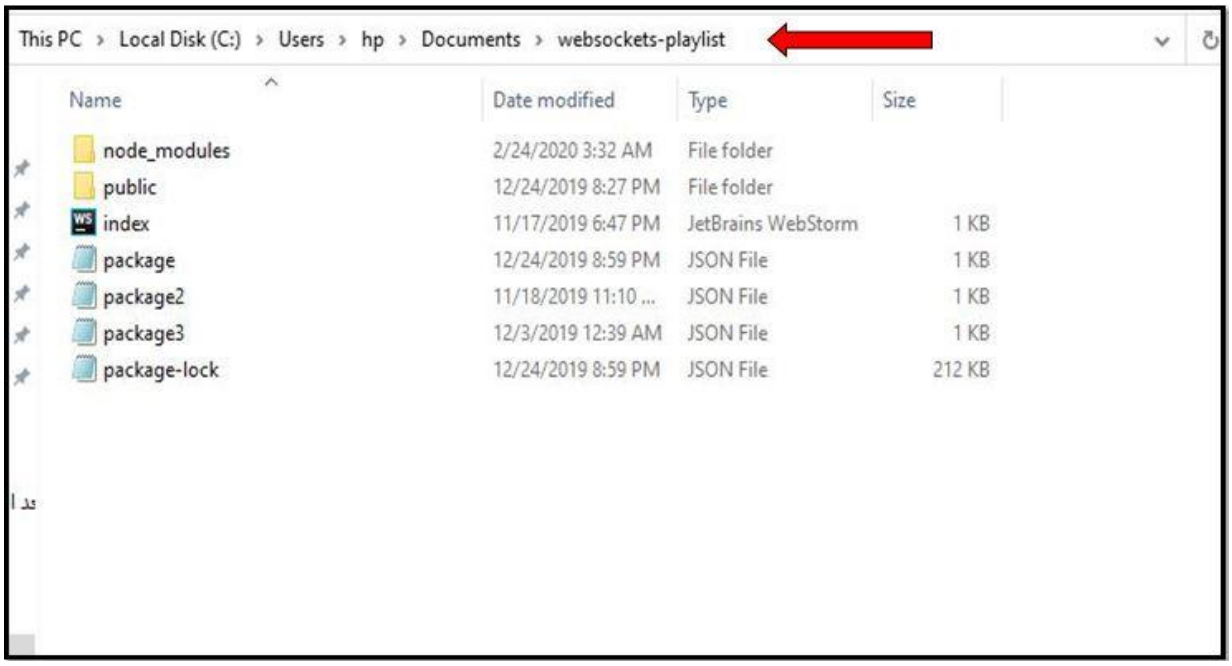


Figure 4.1: The Path of the Program

Then we open (Command Prompt) (an executable file named **cmd**.exe) that is a command-line interpreter available on Microsoft as shown in Figure 4.2.

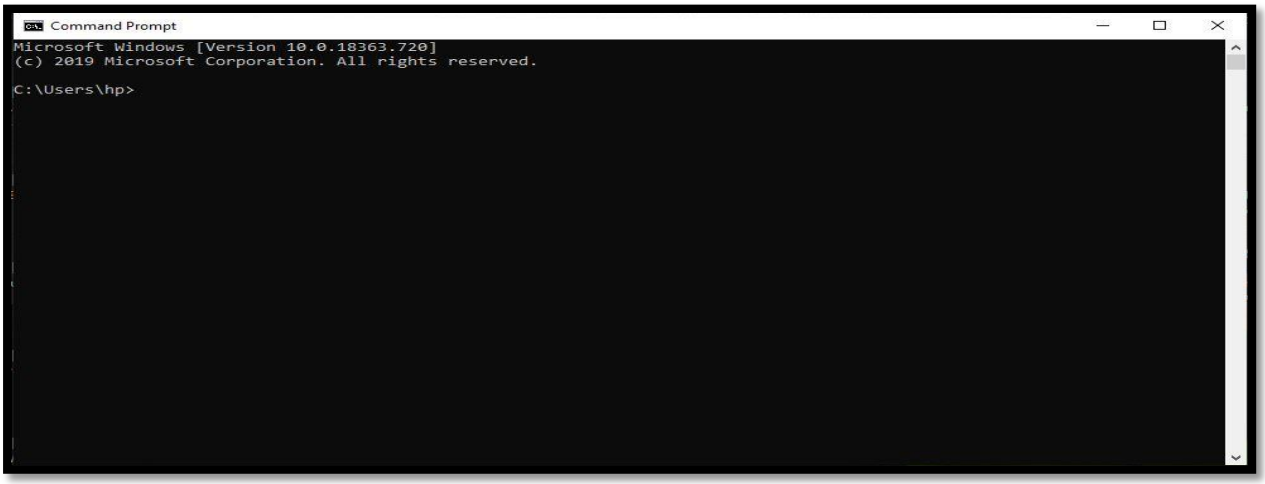


Figure 4.2: The Command Prompt

After that write command “cd” which is a shortcut to the change directory, after that, we make a paste for the path above as shown in Figure 4.3.

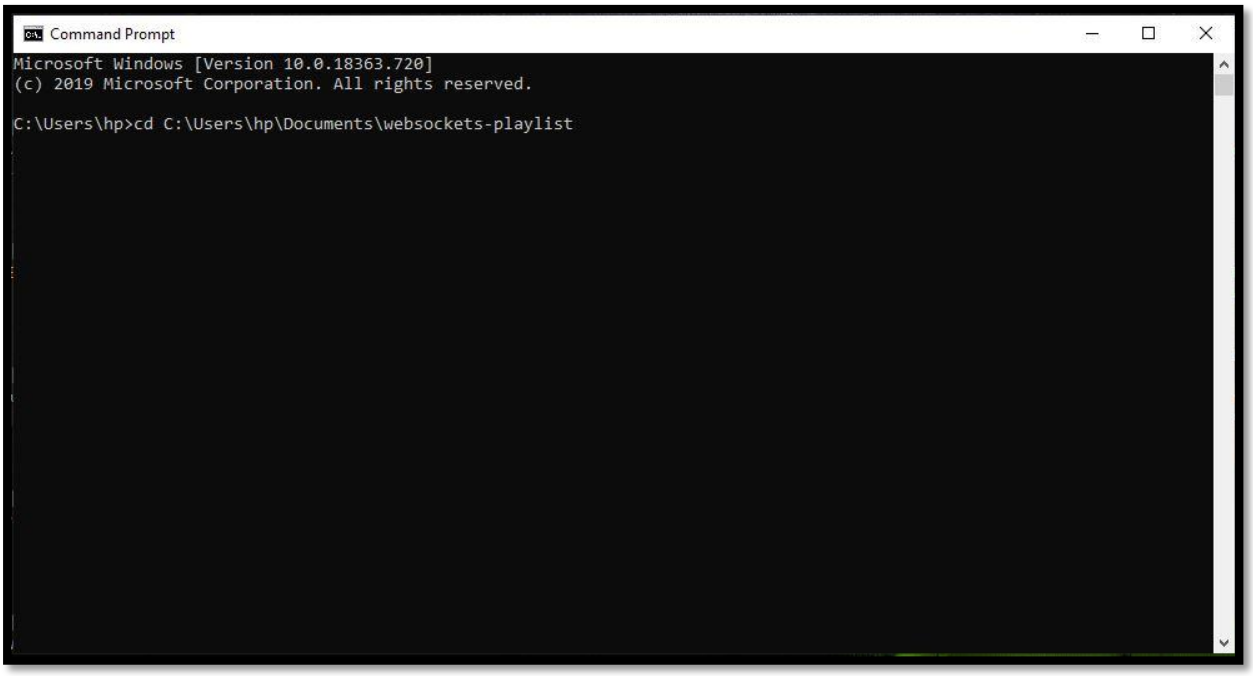


Figure 4.3: Running the command "cd" with the path of web Sockets-playlist

After that, running the command "npm run serve", where the program will run, as shown in Figure 4.4.

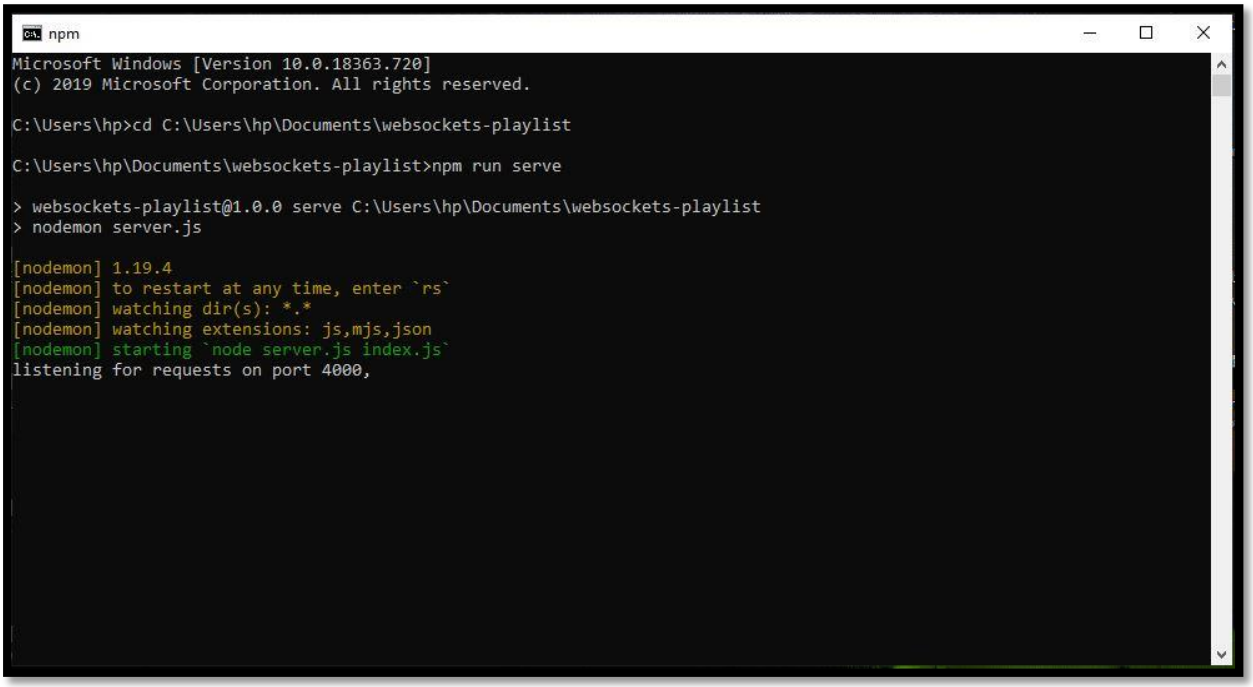


Figure 4.4: Running the Command "npm run serve"

When we going to use the model and make the server running, we open any browser and enter the following link http://localhost:4000/agentAuth.html then click on "Enter" and the home page of the model has been entered as shown in Figure 4.5.

Figure 4.5: Home Page of AgentAuth

When the users or clients want to login to the electronic library, they enter the email and password and click on the login as shown in Figure 4.6.

Figure 4.6: The Login process.

If the user is not registering to the model previously, The user will press sign up as shown in Figure 4.7.

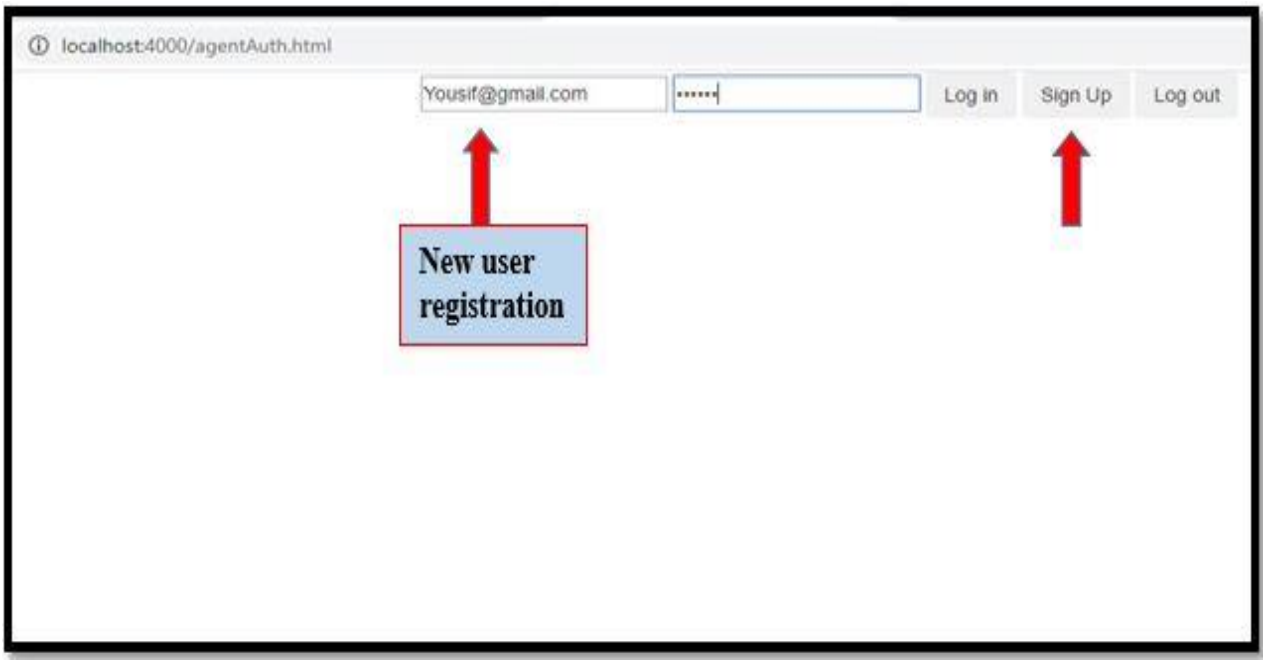


Figure 4.7: The Registry Process for a new user

At this stage, we have made a handshake between the browser and the server, which means that we have entered the agent based on the WebSocket, any user who connects to the model will notice all the communications because it is open communication to everyone as shown in Figure 4.8.

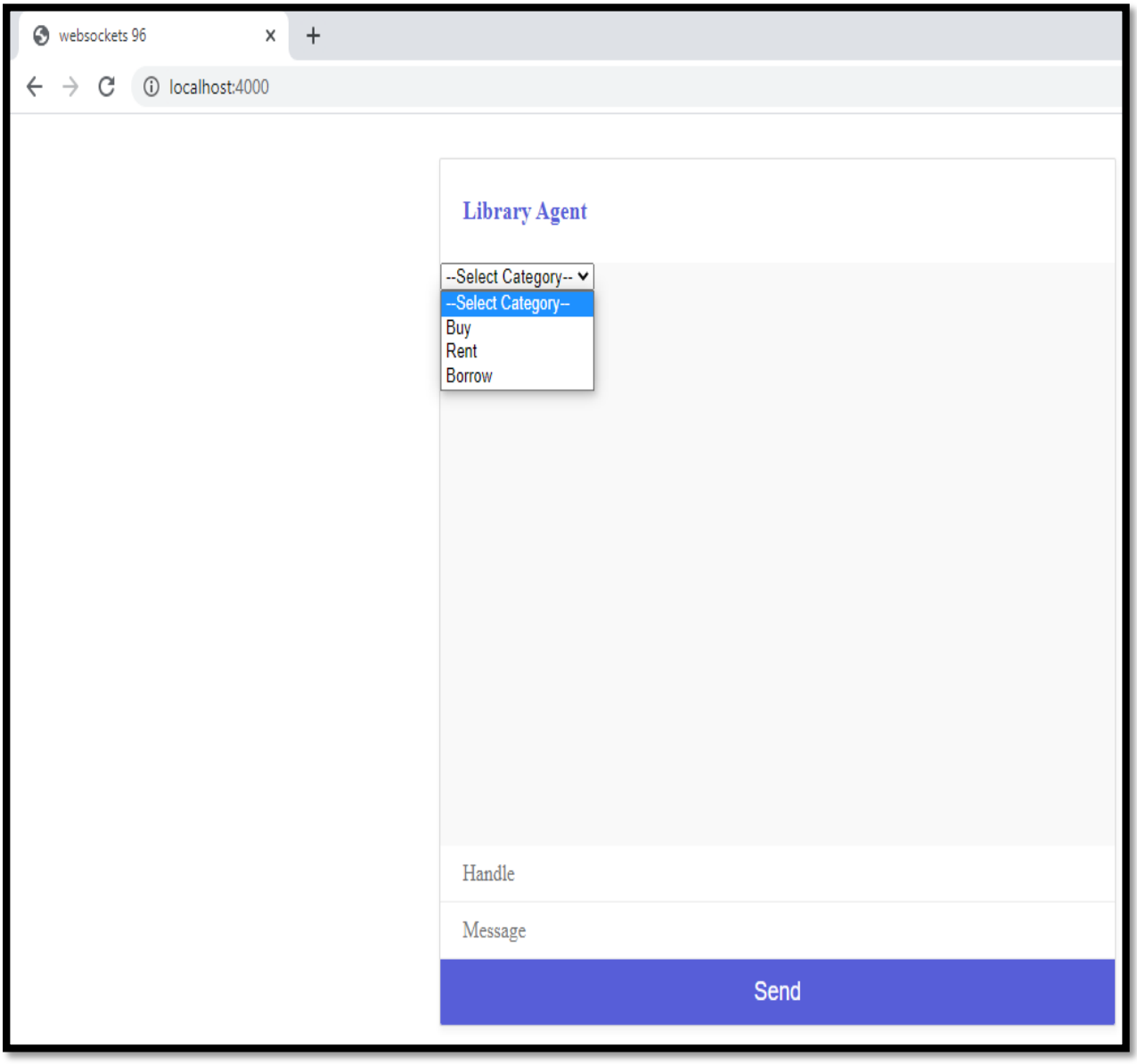


Figure 4.8: Connect to the Library

After entering the library, they select the category: There are three categories: buy, rent, and borrow. After selecting the category, we write the email in the field (handle) and the name of the book in the field (message) and click send as shown in Figure 4.9.

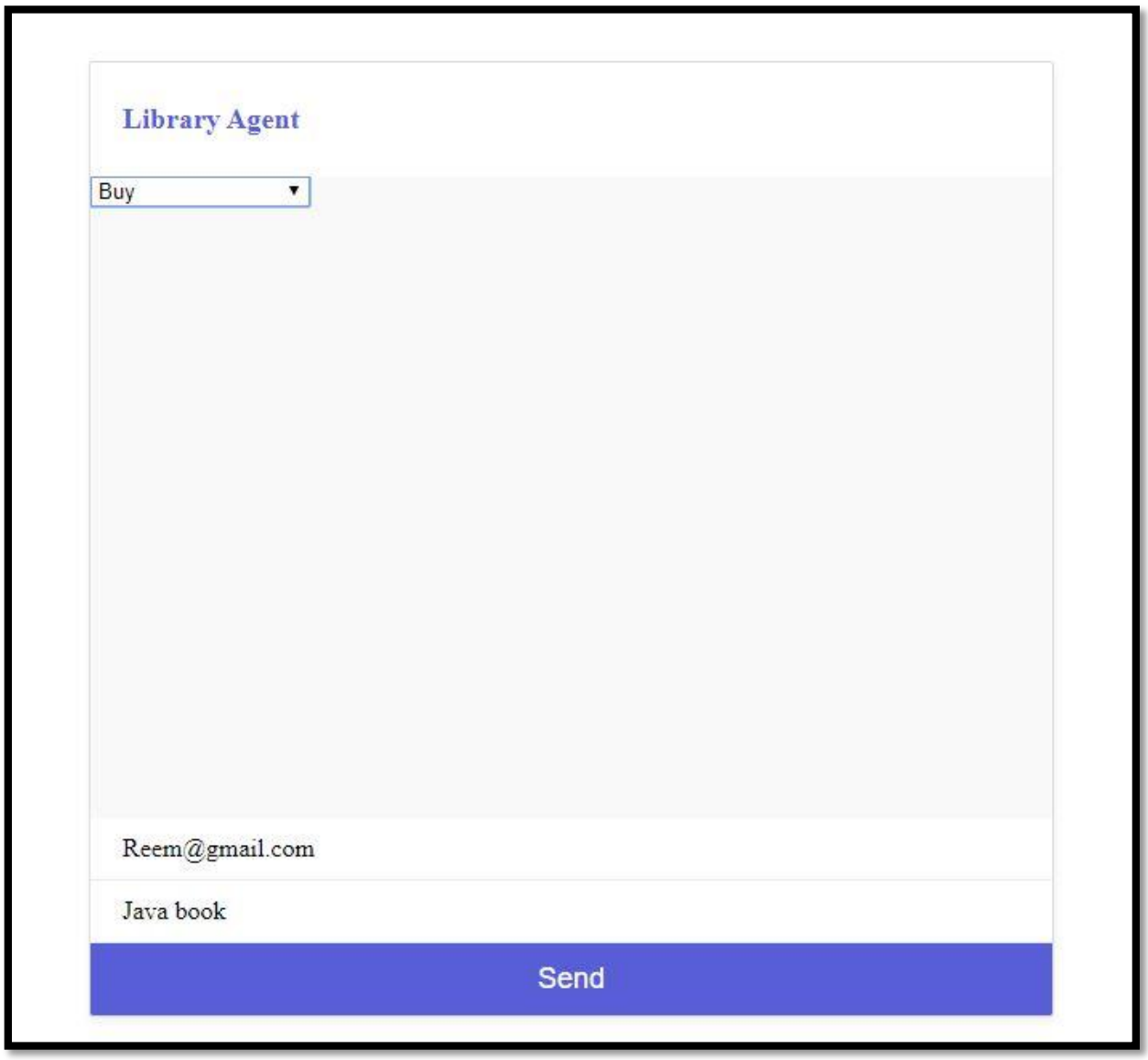


Figure 4.9: The process of ordering a book.

In Figure 4.10 clarified the synchronization process between users or clients who are connected to the model.

Figure 4.10: The asynchronous model with another user.

The second step is sending the request by the user, is shown in Figure 4.11.

Figure 4.11: The second step of asynchronization process

In the third step, the second user or agent response to the request of the first user is shown in Figure 4.12.

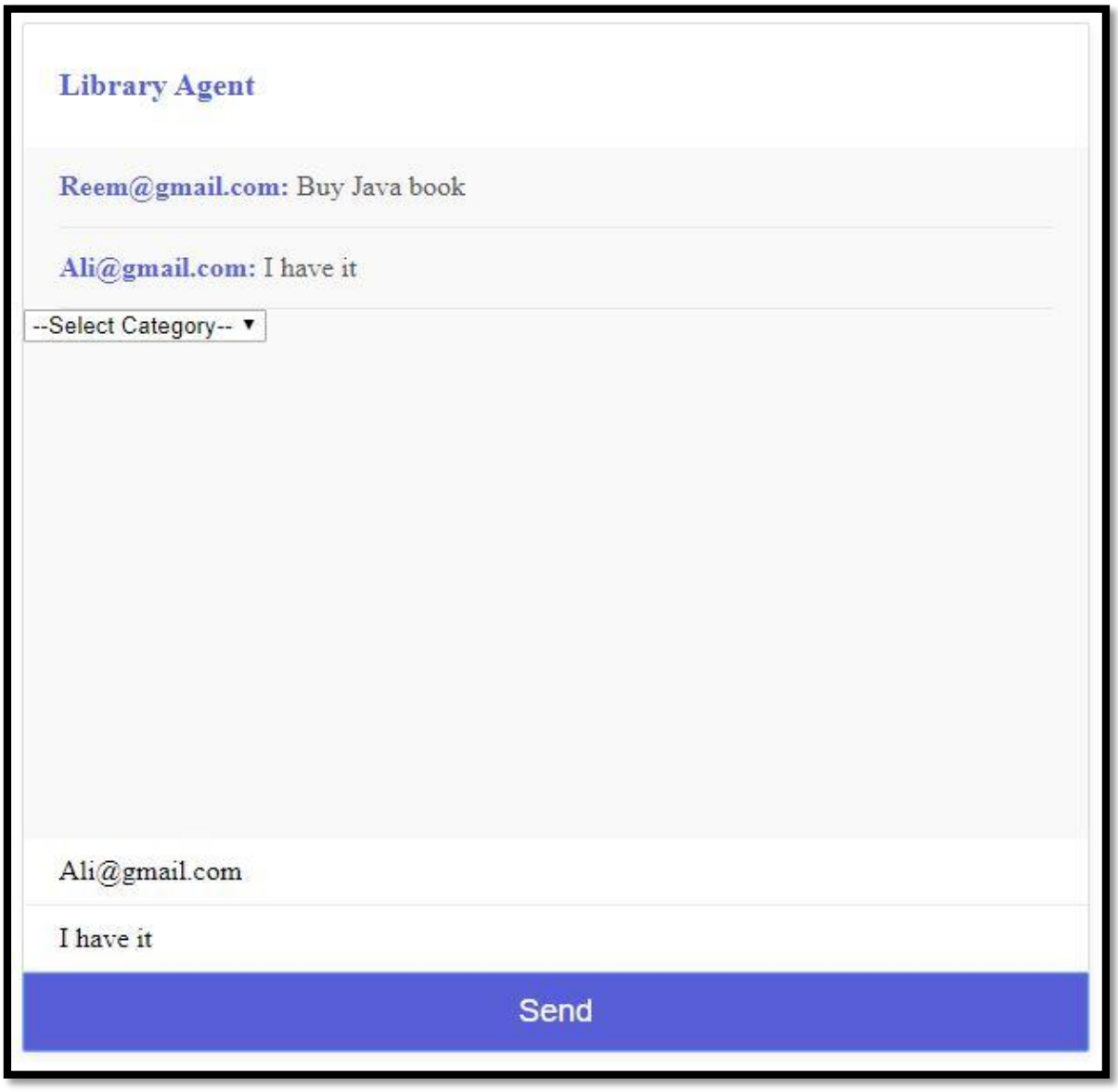


Figure 4.12: The Third step of asynchronization process

When the first user sends the request and the agent library or second user response that the book is in his possession, the library administrator or second user sends the book to the email of the first user who used it to enter the library.

4.3 Visual comparison between HTTP and WebSocket proposed model:

In this section, a comparison of the proposed model that uses the WebSocket and the Web server based on main features. Table (4.1) illustrated comparison of the proposed model that uses WebSocket and web server.

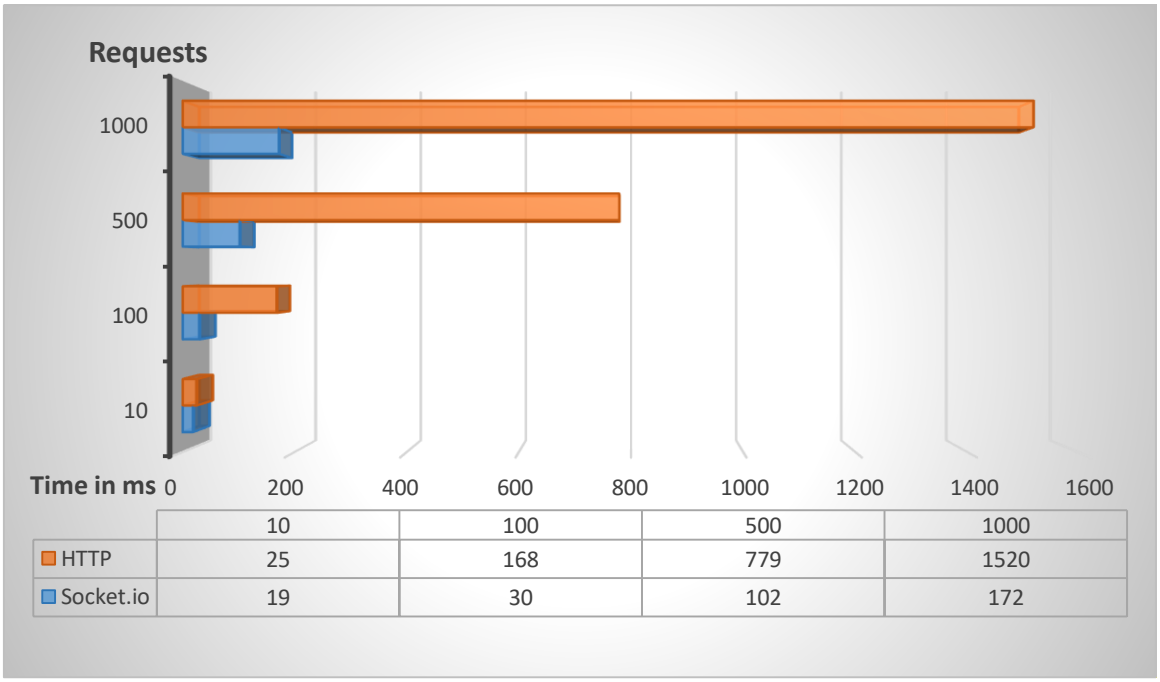
Table (4.1): Compares the proposed model that uses the WebSocket and the Web Server.

used web socket on The proposed model	Web Server
Duplex	
Full duplex	Half duplex
Messaging Pattern	
Bi-direction	Request - Response
Services push	
core feature	Not natively supported. Client polling or streaming download technique used.
Supported Clients	
Modern languages & clients	Broad support
Overhead	
Moderate overhead to establish& maintain the connection, then minimal overhead per message.	Moderate overhead per request / connection.
Intermediary/ Edge caching	
Not possible	Core feature

4.4 The proposed model efficiency:

The time required to implement the proposed model is very fast and is measured in a millisecond, depending on the server being used and the speed of the network. If the server is very big, then it will be measured in a nanosecond with very fast internet that can be measured in a super nanosecond.

In average 10 requests in HTTP take 25 ms and in Socket.io take 19 ms while 100 requests in HTTP need 168 ms and in Socket.io take 30 ms, as well, 500 requests in HTTP take 779 ms and in Socket.io take 102 ms, in addition, 1000 requests in HTTP take 1520 ms and in Socket.io need 172 ms. As is clear from figure 4.13 for our use case WebSocket is expected to be about 5-7 times faster than plain HTTP.



	10	100	500	1000
HTTP	25	168	779	1520
Socket.io	19	30	102	172

Figure 4.13: The proposed model used WebSocket are about 5 to 7 times faster

And below is a Cartesian chart of requests as shown in Figure 4.14.

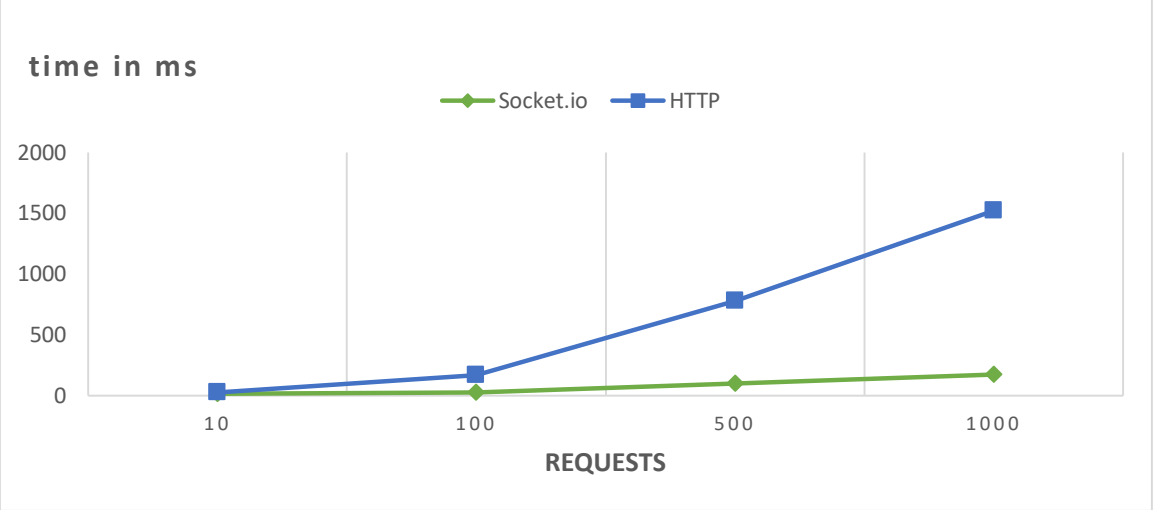


Figure 4.14: A Cartesian chart of requests

4.5 Performance of Proposed Model:

4.5.1 Data transfer

Another interesting number to look at was the amount of data transferred between both protocols. Once established, a WebSocket connection does not have to send headers with its messages so we can expect the total data transfer per message to be less than an equivalent HTTP request.

One HTTP request and response took a total of 282 bytes while the request and response WebSocket frames weighed in at a total of 54 bytes (31 bytes for the request message and 23 bytes for the response). As shown in Figure 4.15.

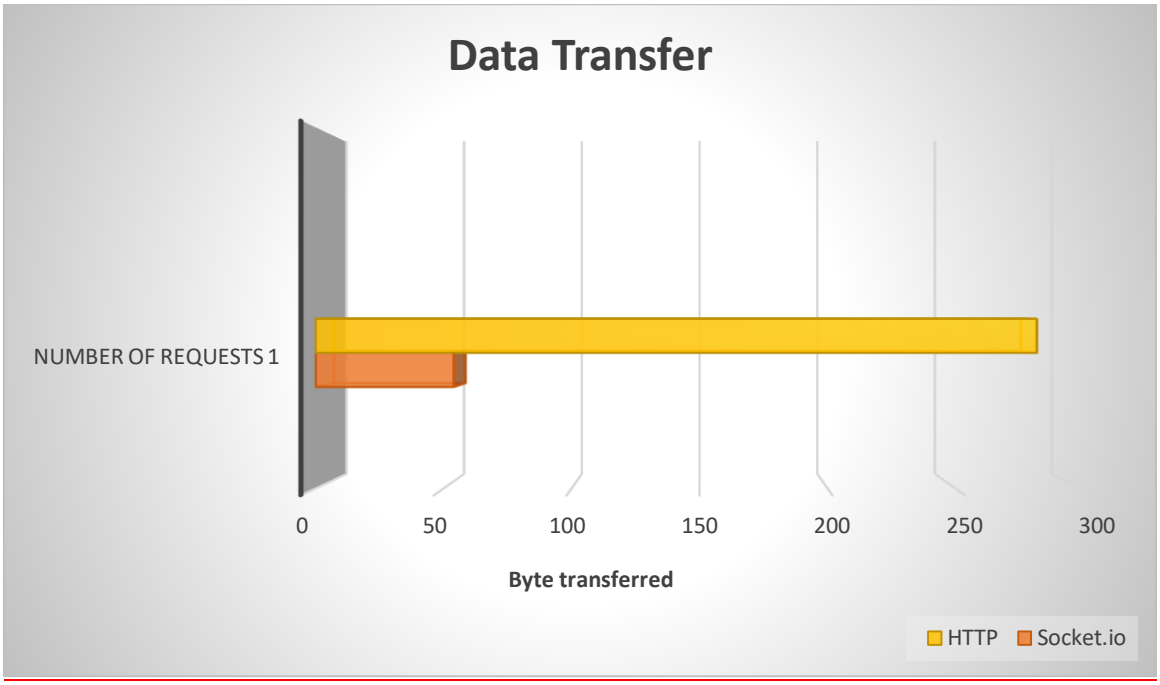


Figure 4.15**:** Data Transfer for one request

Figure 4.16 shows the Cartesian chart of one request

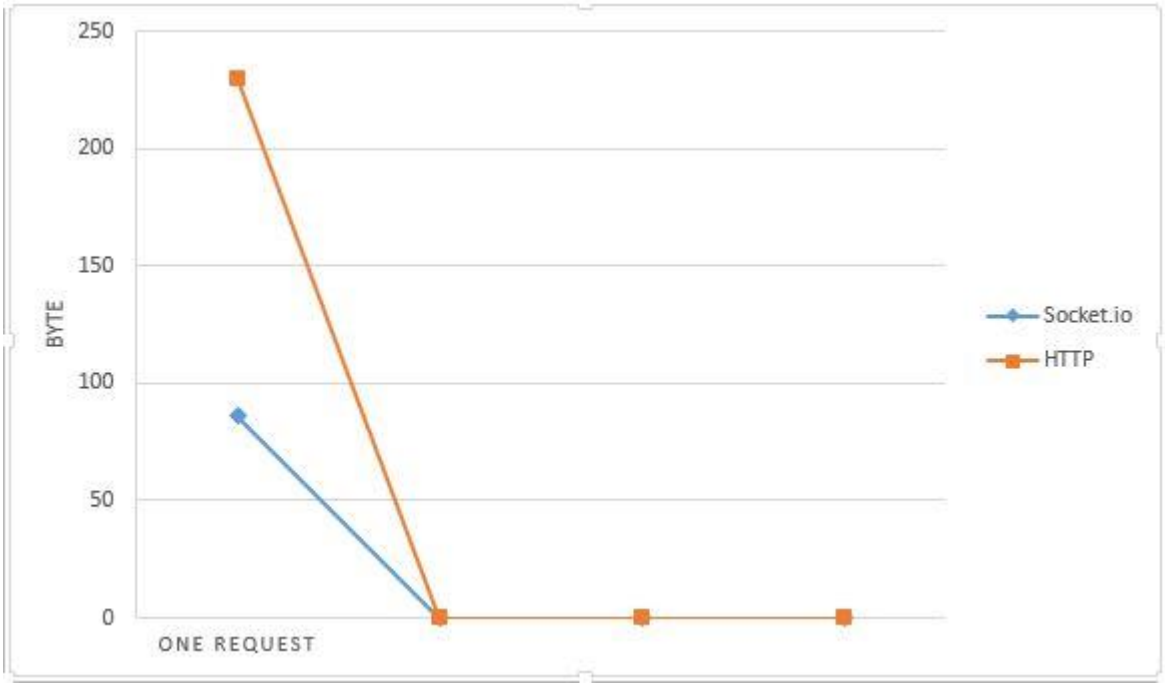


Figure 4.16: A Cartesian chart of one requests

4.5.2 Load benchmarks

The benchmarks in Figure 4.17 are each running 100 concurrent connections include the time it takes to establish the WebSocket connection. Here are the results for 1, 10, and 50 requests per connection:

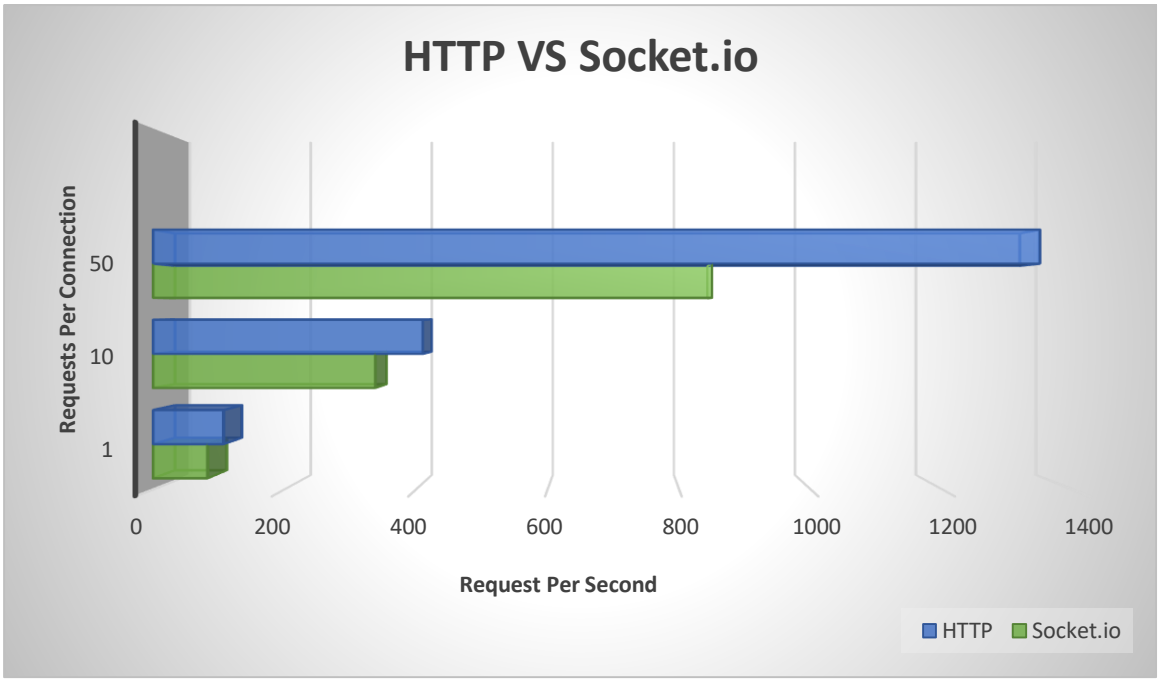


Figure 4.17: Benchmark results for 100 concurrent connections and 1, 10 and 50 requests each

And Cartesian chart in figure 4.18

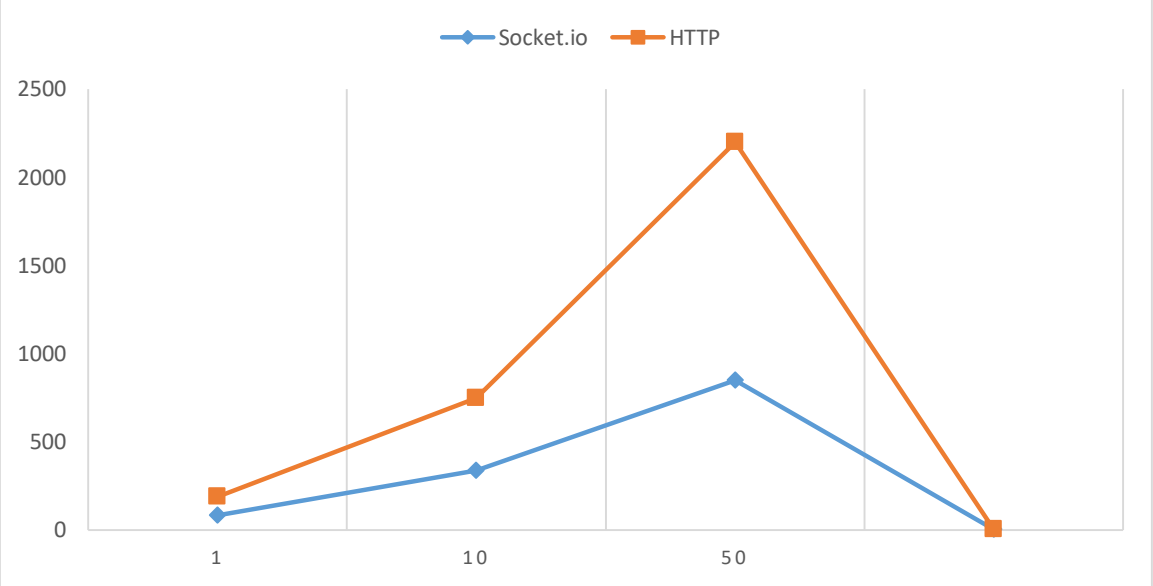


Figure 4.18: A Cartesian chart of requests in Load benchmarks

As we can see in Figure 4.19, making a single request per connection is about 50% slower using Socket.io since the connection has to be established first. This overhead is smaller but still noticeable for ten requests. At 50 requests from the same connection, Socket.io is already 50% faster. To get a better idea of the peak throughput were taken the same benchmark with a larger number (500, 1000 and 2000) of requests per connection:

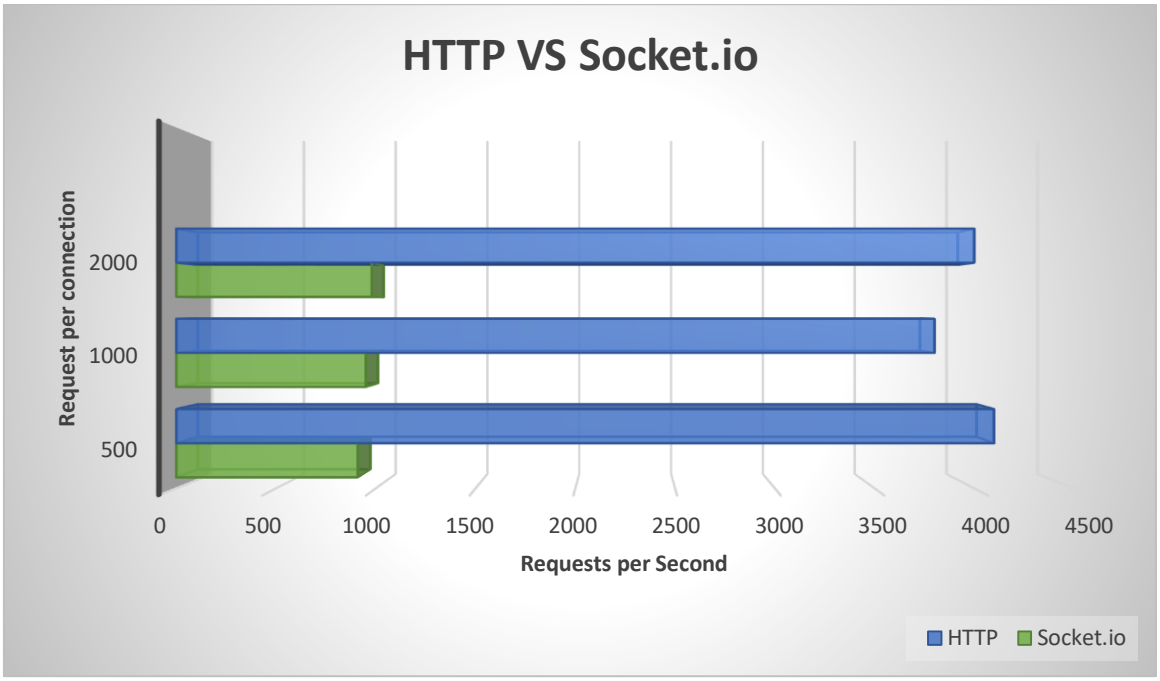


Figure 4.19: Benchmark results for 100 concurrent connections and 500 to 2000 requests each

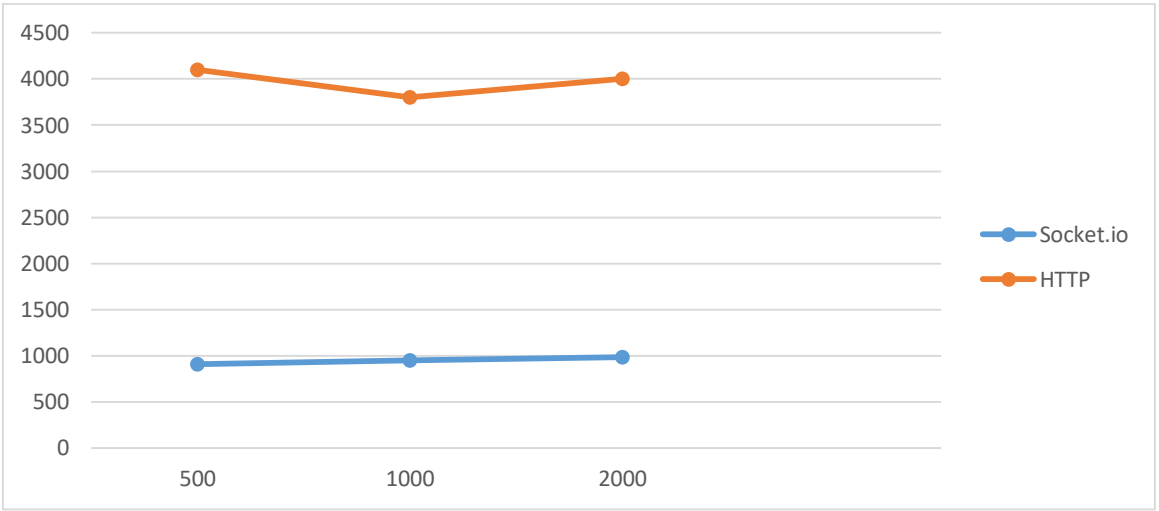


Figure 4.20: A Cartesian chart of requests in Load benchmarks from 500 to 2000 requests

4.6 The reason for not applying the proposed model in practice in the Central Library of Diyala University or the College of Science Library in particular:

The central library needs the following basic requirements:

I. Manpower:

The e-libraries requirement a number of human powers that are featured by having the capacity to work or management information systems on a database (firebase) and automated computers based the Internet during the process of operating, lifting, and save them, including the capacity of these forces to evolve and, activate the e-library from time to time depending on to technological development.

II. Devices:

The devices are the physical components that a computer contains, such as the memory utilized to Store and store information, images, files, programs, and attachments in a Recoverable and permanent way. Devices have to also contain input tools such as the mouse, keyboard, and camera the output tools such as the printer and, monitor, etc.

III. The software:

It is a set of instructions and orders that the device carried out to realize a set of aims and objectives that the human forces lookup.

IV. The electronic library:

The system of university and, college of science does not contain an electronic library to implement the model on it.

V. Electronic sources:

Electronic sources such as books, Research papers, theses, and dissertations are not available enough to be uploaded to the database (Firebase).

VI. The big server:

A large size server must be available to place the form on it. In fact, a large size server permits a faster data transfer process.

In addition, even if all of the above tools are available, crown disease (COVID-19), curfew, and university closures are the biggest cause. For these reasons, we talked about the model only in the theoretical aspect.

Chapter Five

Conclusions and Suggestions for Future works

Chapter Five
Conclusions and Suggestions for Future Works

5.1 Introduction:

In this chapter, we will explain the most important conclusions that we reached from this proposed model and how it is faster and more accurate than other techniques.

5.2 Conclusions:

Several conclusions have been deduced from the obtained test results. In the following some of our conclusions are listed:

1- Using firebase as a database.

2- Provide centralization, asynchronization, and Speed in transferring information using WebSocket agent.

3- The model works in a multi-environment (UNIX, Windows, and Mac or any other operating system, and WebSocket agents using in the present work support different browsers (Chrome, Firefox, and internet explorer) and any languages.

4- WebSocket can determine who can make access to the agent library.

The agent broadcasts the request to all the users or clients' who are existing online on the library.

5- The agent deals with distributed systems, specifically the web application and database application.

The Agent services in distributed systems and web applications are more complex but it is very fast due to an ongoing update.

6- A fast and continuous communication process is achieved until the browser is closed by the user or the computer is turned off or the server is off, then the WebSocket will be disconnected.

The use of WebSocket in the libraries not used before in Iraq and Arabian Countries.

5.3 Future Works:

1- Using Multi-agent in the e-library with adding more details for quick search. It can work as a Ph.D. dissertation because it takes a lot of time.

2- Given the abundance of breakthroughs in the field of the Internet and electronic governance, a smart electronic library can be designed using the blockchain algorithm to maintain information security.

References

REFERENCES

[1] R. Kamble and D. Shah, "Applications of Artificial Intelligence in Human Life," *Int. J. Res.*, vol. 6, no. 6, pp. 178–188, doi: 10.5281/zenodo.1302459, 2018.

[2] S. Russel and P. Norvig, *Artificial intelligence: a modern approach*. Pearson Education Limited, 2013.

[3] N. Kühl, M. Goutier, R. Hirt, and G. Satzger, "Machine learning in artificial intelligence: Towards a common understanding," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[4] C. Schommer, "An unified definition of data mining," *arXiv Prepr. arXiv0809.2696*, 2008.

[5] I. H. Witten, E. Frank, and M. a. Hall, "Data Mining: Practical Machine Learning Tools and Techniques," Third Edition, 2011.

[6] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *IEEE Conference* on Computer Vision and Pattern Recognition (CVPR), 2016.

[8] Cross Validated, "What is the difference between data mining, statistics, Machine Learning, and AI?" 2014.

REFERENCES

[9] J. R. Koza, F. H. Bennett, D. Andre, M. A. Keane, "Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming," in Artificial Intelligence in Design, 1996.

[10] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*, 3rd Education, Pearson, 2015.

[11] J. Haugeland, Artificial Intelligence: The Very Idea. MIT Press, 1989.

[12] R. Bellman, An Introduction to Artificial Intelligence: Can Computers Think? Boyd & Fraser, 1978.

[13] A. Newell and H. A. Simon,"GPS, a program that simulates human thought,", 1961.

[14] D. McDermott and E. Charniak,"Introduction to artificial intelligence," *Int. J. Adapt.* Control Signal Process, vol. 2, no. 2, pp. 148–149, 1985.

[15] E. Rich and K. Knight, "Artificial intelligence," McGraw-Hill, New, 1991.

[16] A. M. Turing, "Computing Machine and Intelligence," MIND, vol. LIX, no. 236, pp. 433– 460, 1950.

[17] D. L. Poole, A. Mackworth, R. G. Goebel,"Computational Intelligence and Knowledge," Comput. Intell. A Log. Approach, no. Ci, pp. 1–22, 1998.

[18] (F. T. Alotaiby, *A component-based functional model for e-learning systems*. George Mason University, 2005.

REFERENCES

[19] A. Pannu, M. Tech, "Artificial Intelligence and its Application in Different Areas", International Journal of Engineering and Innovative Technology (*IJEIT*) Volume 4, Issue 10, P 6, 2015.

[20] R. Evans, "Software Agents : A review 27 May 1997," no. May, 1997.

[21] S. Russell and P. Norvig, *Artificial intelligence: a modern approach.*, Prentice-Hall, *Inc*, 1,152 P, 1995.

[22] N. R. Jennings and M. J. Wooldridge, "Agent Technology: Foundations, Applications, and Markets", Springer, Berlin, Heidelberg, 1998.

[23] S. Roy, A. Banerjee, P. Ghosh, A. Chatterjee, and S. Sen, "Intelligent Web Service Searching Using Inverted Index," in *Contemporary Advances in Innovative and Applicable Information Technology*, Springer, 2019, pp. 13–21.

[24] I. Sommerville, Software Engineering, 7th edition, Addison-Wesley, 2004.

[25] W.-T. Tsai, R. Paul, L. Yu, A. Saimi, and Z. Cao, "Scenario-based web services testing with distributed agents," *IEICE Trans. Inf. Syst.*, vol. 86, no. 10, pp. 2130–2144, 2003.

[26] S. Ghosh and A. Mathur, "Issues in Testing Distributed Component-Based Systems," in Proc. of the First ICSE Workshop on Testing Distributed Component-Based Systems, 1999.

REFERENCES

[27] W. T. Tsai, L. Yu and A. Saimi, "Scenario-Based Object Oriented Test Framework for Testing Distributed Systems," *In Proc*. of the *Ninth IEEE Workshop* on Future trends of Distributed Computing Systems, FTDCS'03, 2003, pp. 288-294.

[28] B. Long and P. Strooper, "A Case Study in Testing Distributed Systems," *in Proc.* of the Third International Symposium on Distributed Objects and Applications, 2001,pp. 20-29.

[29] C. Schommer, "An unified definition of data mining," *arXiv Prepr. arXiv0809.2696,* 2008.

[30] S. Sharples, V. Callaghan, and G. Clarke, "A multi-agent architecture for intelligent building sensing and control," *Sens. Rev.,* 1999.

[31] H. F. El Yamany, M. A. M. Capretz, and L. F. Capretz, "A multi-agent framework for testing distributed systems," in *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, 2006, vol. 2, pp. 151–156.

[32] X. Bai, G. Dai, D. Xu, and W.-T. Tsai, "A multi-agent based framework for collaborative testing on web services," in *The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06)*, 2006, pp. 6-pp.

[33] Y.F. Ma, H. X. Li, and P. Sun, "A lightweight agent fabric for service autonomy," in *International Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering*, 2007, pp. 63–77.

REFERENCES

[34] E. Samanidou, E. Zschischang, D. Stauffer, and T. Lux, "Agent-based models of financial markets," *Reports Prog. Phys.*, vol. 70, no. 3, pp. 409–450, 2007, doi: 10.1088/0034-4885/70/3/R03.

[35] G. Liu, "Scholarship at UWindsor The application of intelligent agents in libraries: a survey Intelligent agents in libraries The Application of Intelligent Agents in Libraries: A Survey," vol. 45, pp. 78–97, 2011.

[36] D. Calvaresi, A. Dubovitskaya, J. P. Calbimonte, K. Taveter, M. Schumacher, "Multi-Agent Systems and Block chain: Results from a Systematic Literature Review", https://www.researchgate.net/publication/324444516, P16, 2018.

[37] S. A. Khan and R. Bhatti, Semantic Web and ontology-based applications for digital libraries, 2018, Emerald Insight, P18.

[38] J. M¨uller, T. Meuser, R. Steinmetz, M. Buchholz, A Trust, "Management and Misbehaviour Detection Mechanism for Multi-Agent Systems and its Application to Intelligent Transportation Systems", eess.SP, P 7, 2019.

[39] A. Bottino and E. Battegazzorre, *"Development Of A Library Of Intelligent Agents For Vr Applications*," 2019.

[40] S. Dilek, H. Cakır, and M. Aydın, "Applications of Artificial Intelligence Techniques to Combating Cyber Crimes: A Review," *Int. J. Artif. Intell. Appl.*, vol. 6, no. 1, pp. 21–39, 2015, doi: 10.5121/ijaia.2015.6102.

[41] M. Šlapák, "Adaptive Control Algorithm of Intelligent Agents", PhD, the Faculty of Information Technology, Czech Technical University in Prague,2019, P 133.

REFERENCES

[42] S. Russel and P. Norvig, *Artificial intelligence: a modern approach*. Second Edition, Prentice Hall Series, 2003, P 113

[43] G. Imre, G. Mezei, "Introduction to a Web Socket benchmarking infrastructure", Zooming Innovation in Consumer Electronics International Conference (*ZINC*), 2016.

[44] I. Fette, the Web Socket Protocol, A. Melnikov Isode Ltd, P 58, 2011.

[45] J.-P. Erkkilä, "Websocket security analysis," *Aalto Univ. Sch. Sci.*, pp. 2–3, 2012.

[46] R. R. Ganji, M. Mitrea, B. Joveski, and F. Preteux.HTML5 as an application Virtualization tool. In Consumer Electronics (ISCE), 2012 IEEE 16th International Symposium on, pages 1 –4, June 2012.

[47] Internet Engineering Task Force (IETF).The Web-Socket Protocol, RFC 6455.

[48] World Wide Web Consortium W3C. The WebSocket API: W3C Candidate Recommendation. http://www.w3.org/TR/websockets/. [Online; received 25 Sep 2012].

[49] O. Cassetti and S. Luz.The WebSocket API as supporting technology for distributed and agent-driven data mining.

http://www.scss.tcd.ie/~casseto/NGDM11-websockets.pdf, 2011. [Online; received 22 Sep 2012].

[50] A. Wessels, M. Purvis, J. Jackson, and S. Rahman. Remote Data Visualization through Web Sockets. In Information Technology: New Generations (ITNG), 2011 Eighth International Conference on, pages 1050 –1051, April 2011.

[51] G. Nicolas, K. Sbata, and E. Najm. Websocket en abler: achieving IMS and web services end-to-end convergence. In *Proceedings of the 5th International Conference on Principles, Systems and Applications of IP Telecommunications*, IPTcomm '11, pages 3:1–3:3, New York, NY, USA, 2011. ACM.

[52] A. H. Barea, "Analysis and evaluation of high performance web servers," p. 116, 2011, doi: 10.1080/03075070020030698.

[53] K. Nørmark, "Using lisp as a markup language–the laml approach," in *European Lisp User Group Meeting*, 1999.

[54] R. Jeyshankar and B. R. Babu, "Websites of universities in Tamil Nadu : a webometric study," vol. 56, no. June, pp. 69–79, 2009.

REFERENCES

[55] N. Kumari, "Web-Based Services in Library and Information Science," *Ijnglt*, vol. 2, no. 1, pp. 1–8, 2016.

[56] L. Liu, K. P. Logan, D. A. Cartes, and S. K. Srivastava, "Fault detection, diagnostics, and prognostics: software agent solutions," *IEEE Trans. Veh. Technol.*, vol. 56, no. 4, pp. 1613–1622, 2007.

[57] V. F. Dent, "Intelligent Agent Concepts in the Modern Library," no. June, 2016, doi: 10.1108/07378830710735894.

[58] M. Moore, T. Ahmed, and A. Glazer, "Using an automated knowledge agent for reference and customer service," *J. Med. Libr. Assoc.*, vol. 92, no. 2, p. 271, 2004.

[59] L. Zick, "The work of information mediators: A comparison of librarians and Intelligent software agents", First Monday, Vol. 5 No. 5, pp. 1-14, 2000, Available at: http://firstmonday.org/issue5_5/zick/index.html.

[60] B. A. Nardi and V. O'day, "Intelligent agents: what we learned at the library," *Libri*, vol. 46, no. 2, pp. 59–88, 1996.

[61] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowl. Eng. Rev.*, vol. 10, no. 2, pp. 115–152, 1995.

[62] L. B. Gasser and C. Braganza, "C. & Hermann, N.(1987). MACE: A Flexible Testbed for Distributed AI Research," *Distrib. Artif. Intell.*, pp. 119–152.

REFERENCES

[63] P. Wavish and M. Graham, "Roles, skills and behaviour: a situated action approach to organising systems of interacting agents," in *International Workshop on Agent Theories, Architectures, and Languages*, 1994, pp. 371–385.

[64] G. T. Payne and O. V Petrenko, "Agency Theory in Business and Management Research," in *Oxford Research Encyclopedia of Business and Management*, 2019.

[65] S. Kemp, "Digitnal in 2018: World's internet users pass the 4 billion mark," 2018. [Online]. Available: https://wearesocial.com/blog/2018/ 01/global-digital-report-2018.

[66] A. Deveria. (2018) Can i use... support tables for html5, css3, etc. [Online]. Available: http://caniuse.com/

[67] P. Eckersley, "How unique is your web browser?," in *International Symposium on Privacy Enhancing Technologies Symposium*, 2010, pp. 1–18.

[68] D. Flanagan, *JavaScript: The Definitive Guide: The Definitive Guide*, no. January 1997. 2006.

[69] L. Moroney, Moroney, and Anglin, *Definitive Guide to Firebase*. Springer, 2017.

REFERENCES

[70] http://www.itinfotech.in/database/difference-between-firebase-mysql-and-sqlite-database/ 2019.

[71] S. John, "CHAT APP WITH REACT JS AND FIREBASE Department of Information Technology," 2010.

[72] Galloway, J. M.,. A cloud architecture for reducing costs in local parallel and distributed virtualized cloud environments. Ph.D. Thesis,University of Alabama Libraries 2013.

[73] Obrutsky, S., Cloud storage: Advantages, disadvantages and enterprise solutions for business. In *Conference: EIT New Zealand*, 2016.

[74] Diaz, A., and Ferris, C.,Ibm's open cloud architecture. *IBM Corp., Armonk, New York*, 2013.

[75] Marks, E. A., and Lozano, B., *Executive's guide to cloud computing*. John Wiley and Sons, 2010.

[76] Paul M. Muchinsky, Node Web Development , vol. 53, no. 9. 2012.

[77] A. Lombardi, *WebSocket: lightweight client-server communications*. " O'Reilly Media, Inc.," 2015.

REFERENCES

[78] H. Shah, “Node. js challenges in implementation,” *Glob. J. Comput. Sci. Technol.*, 2017.

الخلاصـــــة

تعد الخدمات العالمية للوكيل أونظام متعدد الوكلاء مجال بحث جديد واعد. ومع ذلك ، تم اقتراح العديد من التدابير لإثبات فوائد تقنية الوكيل من خلال دعم الخدمات الموزعة وتطبيق تكنولوجيا الوكيل الذكي في ديناميكيات الويب.

هذه الرسالة عبارة عن تصميم لبناء شبكة دلالية على شبكة الويب العالمية لتعزيز إنتاجية إدارة تطبيقات المكتبة الإلكترونية ، حيث توجد مشكلة يواجهها الباحثون والطلاب ، وهي عملية تبادل الكتب من المكتبات الالكترونية حيث انها تكون بطيئة أو أن المكتبة تحتاج إلى بيانات نظام كبيرة.في هذا العمل تم ايجاد حلاً لهذه المشكلة بأستخدام تقنية الوكيل المستندة الى (WebSocket) ، اذ يمكن لأي مستخدم استخدام هذه المكتبة للحصول على اتصال سريع ومعلومات عالية بالاعتماد على النموذج الموجود داخل المكتبة الالكترونية.

سوف يكون النموذج بسيطاً وصغيرا ، فضلا عن ذلك ، لاتحتاج المكتبة الى موظف مسؤول لادخال المعلومات في قاعدة بيانات المكتبة اذ تم وضعها في (Firebase) وهي قاعدة بيانات قائمة على السحابة تقوم بمزامنة البيانات عبر كل عميل في الوقت الحقيقي ، وتوفر وظائف دون اتصال.

يمكن لأي باحث الوصول إلى النموذج عن طريق تسجيل الدخول. ويتم تثبيت هذا التطبيق على خادم المكتبة المركزية ويحصل كل مستخدم يستخدم هذه المكتبة على نتيجة سريعة وتم إثباتها في العمل الحالي. في المتوسط استغرق 10طللبات (HTTP) حوالي 25 ملي ثانية و (Socket.io) 19 ملي ثانية, بينما استغرق 100 طلب (HTTP) حوالي 168 ملي ثانية بينما (Socket.io) 30 ملي ثانية . وايضا, تم حساب 500 طلب (HTTP) استغرق حوالي 779 ملي ثانية و (Socket.io) 102 ملي ثانية , وكما يستغرق 1000 طلب (HTTP) حوالي 1520 ملي ثانية اما (Socket.io) حوالي 172 ملي ثانية. لذلك, فان النموذج المقترح (WebSocket) اسرع من 5 الى 7 مرات من (HTTP) العادي.وايضا في هذه الرسالة تم حساب عملية نقل البيانات ومعايير التحميل.

جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة ديالى

كليــة العلوم

خدمات الوكيل الذكي في النظام الموزع

رســالة

مقدمة الى مجلس كلية العلوم - جامعة ديالى وهي جزء من متطلبات نيل درجة الماجستير في (علوم الحاسوب)

من قبل

ريم عادل قادر الربيدي

بإشراف

أ. ناجي مطر سحيب

1442هـ العراق 2020م