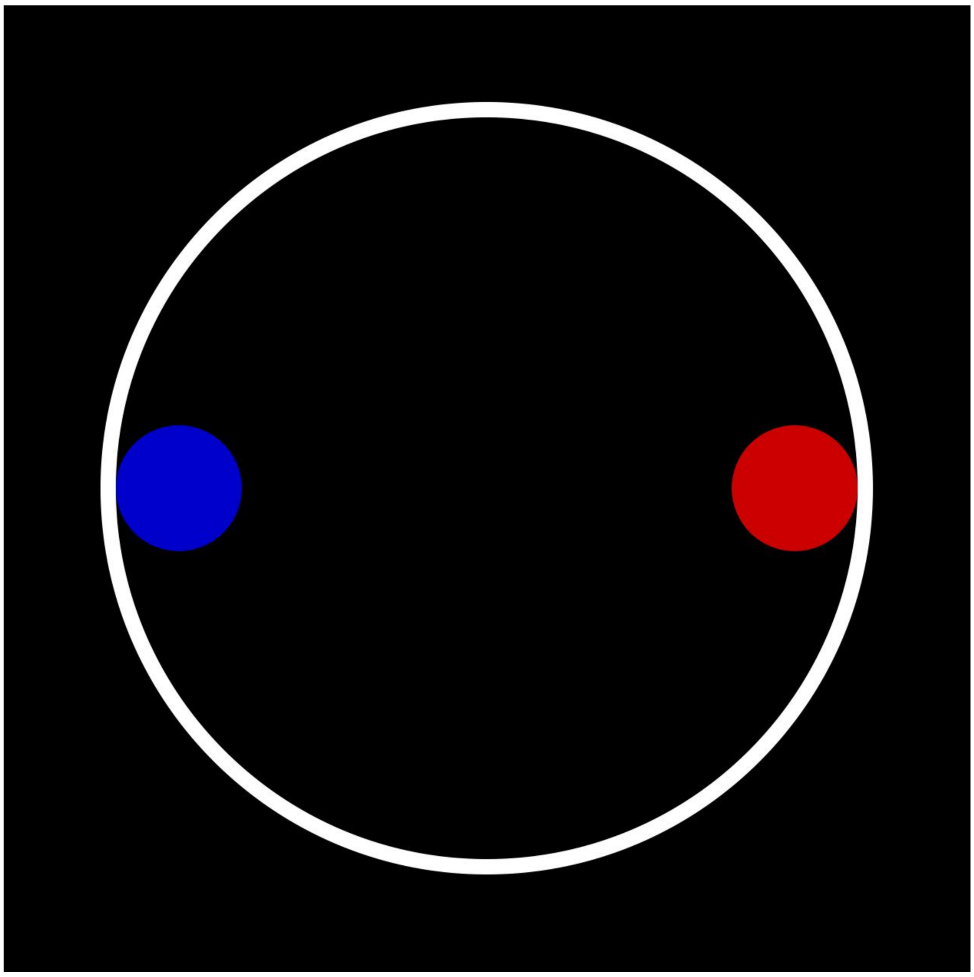
Turing test EXtended Architecture (TEXA):

A method for interacting, evaluating and assessing AIs



Project Report

2016-2017

by

Ganesh Prasad Kumble

(github.com/kggp1995)

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

"Turing test" - A test for intelligence in a computing machine, requiring that a human being should be unable to distinguish the machine from another human being by using the replies to questions put to both. This test designed by Alan Turing in 1950 has faced several criticisms from many experts in the field of AI (Artificial Intelligence) and philosophical thinkers of Artificial Psychology.

However, to interact, evaluate and assess the modern AIs, an enhanced framework is required in order to address a few criticisms, that are relevant to developments observed in the present and projected in the near future. Few criticisms that seem to be relevant in the perspective of applicability of AI in everyday use are:

1. Non quantification of the Machine Intelligence
2. Non quantification of the Human Experience

This project emphasizes on addressing the above 2 major criticisms by introducing a framework backed by a mathematical oriented theory for interacting evaluating and assessing AIs.

Chapter 1

INTRODUCTION

1.1 Introduction

Artificial Intelligence (AI) is defined as the intelligence exhibited by machines. This exhibited feature of a machine is polymorphic in nature and hence, can be observed and adapted in many forms. AI has changed our personal lifestyle and business practices in many ways; and will continue to exhibit more in the coming future. However, with the current rate of developments in the AI, we are rapidly moving towards a state of utopia and hence in a need of a neutral and a common framework to interact, evaluate and assess the potential of these said machines. This initiative was led by Alan Turing (an English computer scientist, mathematician, and well known cryptanalyst during World War II) Et Al., in 1950. The resulting mechanism was termed "Turing test" - A test for intelligence in a computing machine, requiring that a human being should be unable to distinguish the machine from another human being by using the replies to questions put to both.

1.2 Statement of the Problem

The standard and original version of the Turing test currently available was designed by Alan Turing in 1950. This original version has faced several criticisms from many experts in the field of AI and philosophical thinkers of Artificial Psychology:

1. Non quantification of the Machine Intelligence: The Original Turing test doesn't provide a 'Gradient of Intelligence' exhibited by the machine under test. This criticism needs to be addressed in order to qualify an AI based on its functions into sections; and scores into levels.

2. Non quantification of the Human Experience: The original Turing test doesn't provide a 'Open Knowledgebase' of the interrogators who perform and operate the test. This criticism needs to be addressed in order to ensure the worthiness of test as well as the potential of both the Human Intelligence and Artificial Intelligence involved in the instance.

1.2.1 Proposed System:

To interact, evaluate and assess the modern AI, an enhanced framework is required in order to address a few criticisms, that are relevant to developments observed in the present and projected in the near future. The above mentioned criticism seems to be relevant in the perspective of applicability of AI in everyday use.

1.3 Objective of the System:

The main objective of the system is to develop a quantitative model based framework that can mathematically sign each interaction between the Human Interrogator/User & the AI. The framework aims to leverage the collected scores and leverage the same for further assessment and applications.

1.4 Scope of the System:

The scope of the proposed system is limited to providing a mathematical framework for assessing the AIs, implemented via a native platform that felicitates integrated communication i.e., interaction with the AI and evaluating the interaction between the subjected AI. The project doesn't emphasize on development techniques or patterns for AI, but on the evaluation techniques.

1.5 Methodology:

The principle behind the Turing test involves a combination of simple and complex interrogation with the exhibited Artificial Intelligence (AI) in the instance and the human. The mode of interrogation can be of many forms such as: textual, video, picture show and tell, etc. However, the proposed system and its underlying architecture involve measuring the potential of the AI and the Human under interrogation by leveraging the competitive evaluation framework. Hence, the process is divided into 3 discrete phases:

Phase 1: Connecting to the AI exhibiting Host. Log the underlying activity.

Phase 2: Conducting the Interrogation with the exhibiting AI.

Phase 3: Deriving conclusions based on the definitions under the competitive framework.

The competitive evaluation framework is designed in order to quantify the potential of the AI and segregate accordingly. This is made possible by considering various factors mentioned under the Governing Laws of Artificial Psychology and the recommended Guidelines for Human User Experience with Digital entities.

1.6 Related Work:

Although a thorough search was made, we were able to recognize only one similar work:

- *"Quantifying Natural and Artificial Intelligence in Robots and Natural Systems with an Algorithmic Behavioral Test*" by Hector Zenil – A 21 page report published in the Springer Cosmos Series Book on Metrics of sensory motor integration in robots and animals. Although the author emphasizes on the need for quantification, it lacks a common runtime framework for interaction, evaluation and assessment.

1.7 Organization of the Report:

This section is intended to give the reader a brief overview of the structure of this document and the composition of each chapter:

- Chapter 1 contains introduction to the AI contemporary, issues faced and the methodology employed to address the problems.
- Chapter 2 contains a brief overview of all the elements leveraged by the framework.
- Chapter 3 specifies the Software Requirement Specification.
- Chapter 4 consists of the design of this system and its components. It also includes Data flow diagram, flowchart and description about modules.
- Chapter 5 explains the implementation of the module. It also consists of pseudo code and explains how the code works.
- Chapter 6 includes different levels of testing and presents various test cases with its results.

- Chapter 7 provides sample outputs illustrating the working of the application by different actors as snapshots.
- Chapter 8 concludes the work and presents some possibilities for further enhancements in this work.

Chapter 2

LITERATURE SURVEY

2.1 Artificial Intelligence:

Artificial Intelligence (AI) is defined as the intelligence exhibited by machines. This exhibited feature of a machine is polymorphic in nature and hence, can be observed and adapted in many forms. In computer science, the field of AI research defines itself as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of success at some goal. [4]

The overall research goal of AI is to create technology that allows computers and machines to function in an intelligent manner.

AI has undoubtedly changed our personal lifestyle and business practices in many ways; and will continue to exhibit more in the coming future. However, with the current rate of developments in the AI, we are rapidly moving towards a state of utopia and hence in a need of a neutral and a common framework to interact, evaluate and assess the potential of these said machines.

2.2 Human Intelligence:

Human intelligence is the intellectual prowess of humans, which is marked by high cognition, motivation, and self-awareness. Through this intelligence, humans possess the abilities to learn from concepts, understand, and apply logic and reason. [5] The intelligence is not just limited to comprehend nature’s entities but also assists humans in recognizing patterns and helps communicate.

2.3 Turing Test:

A Test methodology in which Turing proposed that a human evaluator would judge natural language conversations between a human and a machine that is designed to generate human-like responses. The evaluator would be aware that one of the two partners in conversation is a machine, and all participants would be separated from one another. The conversation would be limited to a text-only channel such as a computer keyboard and screen so that the result would not be dependent on the machine’s ability to render words as speech. If the evaluator cannot reliably tell the machine from the human (Turing originally suggested that the machine would convince a human 70% of the time after five minutes of conversation), the machine is said to have passed the test. The test does not check the ability to give correct answers to questions, only how closely answers resemble those a human would give. [6]

2.4 ELIZA:

ELIZA is a Computer Program for the Study of Natural Language Communication between Man and Machine. The program was introduced by Joseph Weizenbaum from Massachusetts Institute of Technology. ELIZA makes natural language conversation with a Computer possible. ELIZA was implemented in Mad-SLIP language on an IBM 7094 in the year 1966. Also, ELIZA is the proud winner of the Loebner’s Prize and the first ever to pass the Turing Test. [3]

2.5 Artificial Psychology:

Artificial psychology is a theoretical discipline proposed by Dan Curtis (1963). The theory considers the situation when an artificial intelligence approaches the level of complexity where the intelligence meets two conditions:

Condition I:

A: Makes all of its decisions autonomously

B: Is capable of making decisions based on information that is

1. New
2. Abstract
3. Incomplete

C: The artificial intelligence is capable of reprogramming itself based on the new data

D: And is capable of resolving its own programming conflicts, even in the presence of incomplete data. This means that the intelligence autonomously makes value-based decisions, referring to values that the intelligence has created for itself.

Condition II:

All four criteria are met in situations that are not part of the original operating program.

When both conditions are met, then, according to this theory, the possibility exists that the intelligence will reach irrational conclusions based on real or created information. [7]

2.6 The Chinese Room:

The Chinese Room uses the analogy of a native English speaker with no knowledge of how to speak, write, or read Chinese who is given a couple of sets of rules in English. These rules correlate input in Chinese to coherent output also in Chinese, even though the “translator” only speaks English. This question has been the topic of various research topics in Machine Learning and Natural Language Processing.

2.7 Machine Learning:

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can change when exposed to new data. The process of machine learning is similar to that of data mining. Both systems search through data to look for patterns. However, instead of extracting data for human comprehension -- as is the case in data mining applications -- machine learning uses that data to detect patterns in data and adjust program actions accordingly. Machine learning algorithms are often categorized as being supervised or unsupervized. Supervised algorithms can apply what has been learned in the past to new data. Unsupervised algorithms can draw inferences from datasets.

2.8 Natural Language Processing (NLP):

NLP is a way for computers to analyze, understand, and derive meaning from human language in a smart and useful way. By utilizing NLP, developers can organize and structure knowledge to perform tasks such as automatic summarization, translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation. Apart from common word processor operations that treat text like a mere sequence of symbols, NLP considers the hierarchical structure of language: several words make a phrase, several phrases make a sentence and, ultimately, sentences convey ideas.

NLP is used to analyze text, allowing machines to understand how humans speak. This human-computer interaction enables real-world applications like automatic text summarization, sentiment analysis, topic extraction, named entity recognition, parts-of-speech tagging, relationship extraction, stemming, and more. NLP is commonly used for text mining, machine translation, and automated question answering.

NLP techniques and mechanisms are used in this project extensively to demonstrate an AI pattern exact to ELIZA specification.

2.9 Fuzzy Logic:

Fuzzy logic is an approach to computing based on "degrees of truth" rather than the usual "true or false" (1 or 0) Boolean logic on which the modern computer is based. The idea of fuzzy logic was first advanced by Dr. Lotfi Zadeh of the University of California at Berkeley in the 1960s. Dr. Zadeh was working on the problem of computer understanding of natural language.

Natural language (like most other activities in life and indeed the universe) is not easily translated into the absolute terms of 0 and 1. (Whether everything is ultimately describable in binary terms is a philosophical question worth pursuing, but in practice much data we might want to feed a computer is in some state in between and so, frequently, are the results of computing.) It may help to see fuzzy logic as the way reasoning really works and binary or Boolean logic is simply a special case of it. Fuzzy logic includes 0 and 1 as extreme cases of truth (or "the state of matters" or "fact") but also includes the various states of truth in between so that, for example, the result of a comparison between two things could be not "tall" or "short" but ".38 of tallness." Fuzzy logic seems closer to the way our brains work. We aggregate data and form a number of partial truths which we aggregate further into higher truths which in turn, when certain thresholds are exceeded, cause certain further results such as motor reaction. A similar kind of process is used in neural networks, expert systems and other artificial intelligence applications.

Fuzzy logic is essential to the development of human-like capabilities for AI, sometimes referred to as artificial general intelligence: the representation of generalized human cognitive abilities in software so that, faced with an unfamiliar task, the AI system could find a solution.

2.10 Human-Machine Interaction (HMI):

HMI is a study of interactions between humans and machines. Human Machine Interaction is a multidisciplinary field with contributions from Human-Computer interaction (HCI), Human-Robot Interaction (HRI), Robotics, Artificial Intelligence (AI), Humanoid robots and exoskeleton control.

Go or GoLang is a free and open source programming language created at Google in 2007 by Robert Griesemer, Rob Pike and Ken Thompson. It’s a compiled, statically typed language with garbage collection, and memory safety features. The Go language is extensively used in our project to deliver robust and scalable framework for higher availability of the interaction platform as well as mathematical functionalities.

Chapter 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 Purpose:

The purpose of the software requirement specification is to identify the functional and non-functional requirements necessary for developing the application.

3.2 Scope:

The scope of this project is to deliver a client-server model based framework suitable for initializing a Human-Machine interaction with an AI, evaluate each transaction and assess the characteristics of the AI using the collected scores.

3.3 Operating Environment:

- Linux Environment with any GoLang IDE is suitable for development purposes
- Linux Environment for system deployment
- Linux/Windows/OSX/Android/iOS or any Operating System environment with a browser support for accessing the services of the framework

3.4 Assumptions and dependencies:

- The system is supported in all OS environments supporting GoLang and MEAN Stack.
- Database & Parsers are designed from scratch. Hence doesn't depend on external databases.
- Thus, the system has good scope in all environments, with minimum cost overload.

3.5 Actors and Roles:

- **System/Framework Developer**

  The system developer is responsible for implementing TEXA Theory elements & principles by eliciting new and existing RFCs.

- **AI Developer**

  The AI Developer is responsible for developing the Engine & Language processing data / Dictionary for benchmarking the AI as per the specifications.

- **Interrogator**

  The interrogator is a pseudo-actor responsible for carrying out interaction with a loaded AI and evaluates each interaction with the AI. The evaluation is further used by the framework for standard and customized assessment.

- **Client / End User**

  The end user doesn't involve with any of the internal process mentioned above, but can just derive the calculated results by accessing the system's result datastores trough an interface.

3.6 Functional Requirements:

The project presents the user an interactive and effective environment for uploading, interacting, evaluating and assessing the AI, all under a common platform. The system provides following functionalities:

3.6.1 Uploader Page

- The Uploader feature facilitates the interrogator with a dialog that is linked with the native file explorer to locate and upload the necessary files required to run the AI in the Interrogator.
- The Uploader also provides an opportunity to designate pseudo-names to the AI in order to secure privacy of the AI under development.

3.6.2 Acknowledgement Page

- The acknowledger acknowledges two types of events:
    - In the case of AI file(s) upload, the Acknowledger provides a decent result mentioning the original filename, its associated metadata and the location where the AI data is temporarily stored in the server. Otherwise, resets connection without sending any data back to the interrogator client.
    - In the case of AI Interrogation, the Acknowledger provides an acknowledgement that all the recorder quantum scores evaluated to each transaction are received by the server. Otherwise, resets connection without sending any data back to the interrogator client.

3.6.3 Interrogator Page

- The interrogator facilitates the human-machine interaction.
- The interrogator also provides a facility to add pseudo-names for Interrogators to protect individual privacy.
- The interrogator is responsible for securely recording the user input, transferring the same to the AI Engine, and retrieving the corresponding results.
- The interrogator is also responsible for recording the quantum score assigned to each transaction assigned/decided by the interrogator.

3.6.4 Results Page

- The results page retrieves the assessment data calculated and stored in the native datastore of the framework.
- The results page may be modified by the user to interpret the output in a customized manner or graphically as well.

3.7 Non-Functional Requirements:

- **Portability**

  The system should be able to run on any generic OS & handheld devices. The native implementation supports these aspects very much.

- **Performance**

  The response time should be quick so that either the user or interrogator doesn't have to wait and get disappointed before using the application.

- **Usability**

  The system provides efficient user interface to use it with ease and simple knowledge.

3.8 System Analysis:

3.8.1 Hardware Requirements

Processor	: INTEL Ivy bridge or Xeon Series, AMD A8 or similar
Ram	: Minimum 8GB
Hard Disk	: 1TB or more, NAS and SSD are optional
Compact Disk	: NA
Input device	: Mouse, Keyboard
Output device	: Monitor, Thin clients & Terminals

3.8.2 Software Requirements

Language	: GoLang 1.7 +, GCC, GNU CPP, HTML
Database	: Native JSON Key-Value Store
Front-End	: MEAN Stack - JavaScript enabled browser
Operating System	: Linux (Kernel 4+) / Windows 10 / OSX

Chapter 4

DESIGN

4.1 Design Goals:

The goal of the system is to provide a uni-platform for interacting, evaluating and assessing the AI under exhibit. The embedded User Interface provided with the system, facilitates an easier overall experience.

4.2 Data flow model:

A Data Flow Diagram (DFD) is a graphical representation of the flow of data through an information system, modeling its process aspects. A DFD shows what kind of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of the processes, or information about whether processes will operate in sequence or in parallel.

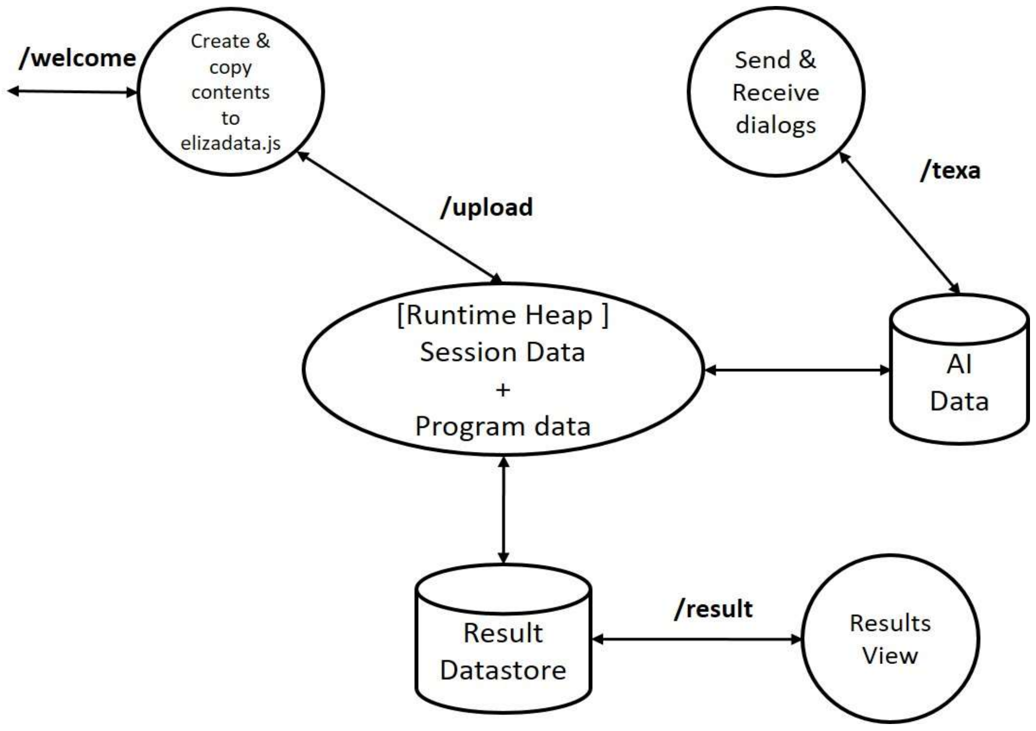


Fig 4.1: Data Flow Diagram of the proposed system

4.3 State Machine Model:

A state machine model describes how a system responds to internal or external events. The state machine model shows system states and events that cause transitions from one state to another. It does not show the data flow within the system. This model is often used in the system which is being driven by the stimuli from the system's environment. A state machine model assumes that, at any time, the system is in one of the number of possible states.

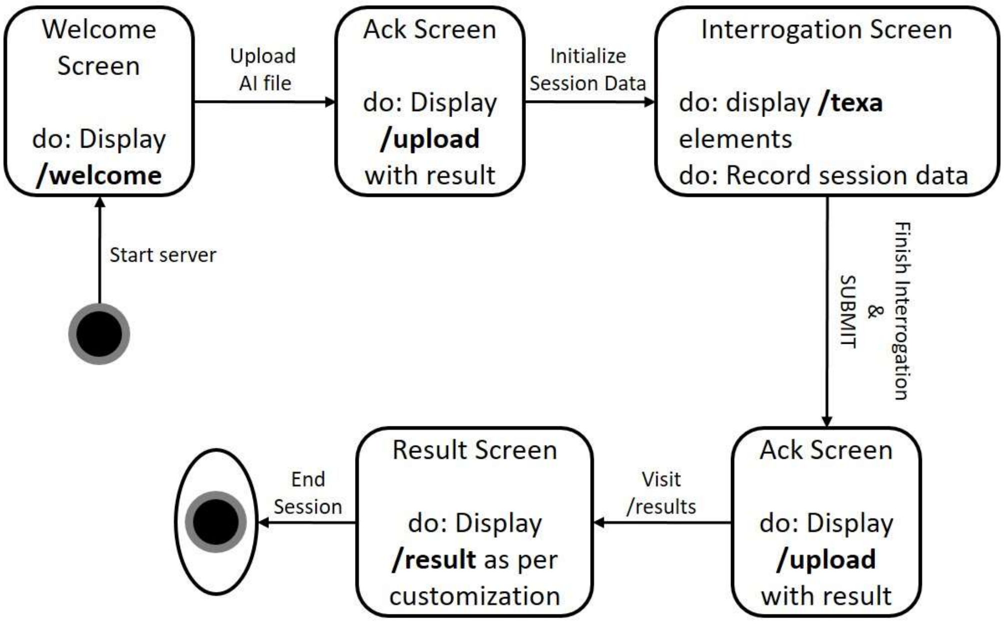


Fig 4.2: State Transition Diagram of the proposed system

4.4 Class Diagram:

A class describes the group of objects with the same attributes and properties, kinds of relationships and semantics. Class diagrams provide a graphic notation for modeling classes and their relationships thereby describing possible objects. Class diagrams are useful both for abstract modeling and for designing programs. The UML Notation for classes are as follows:

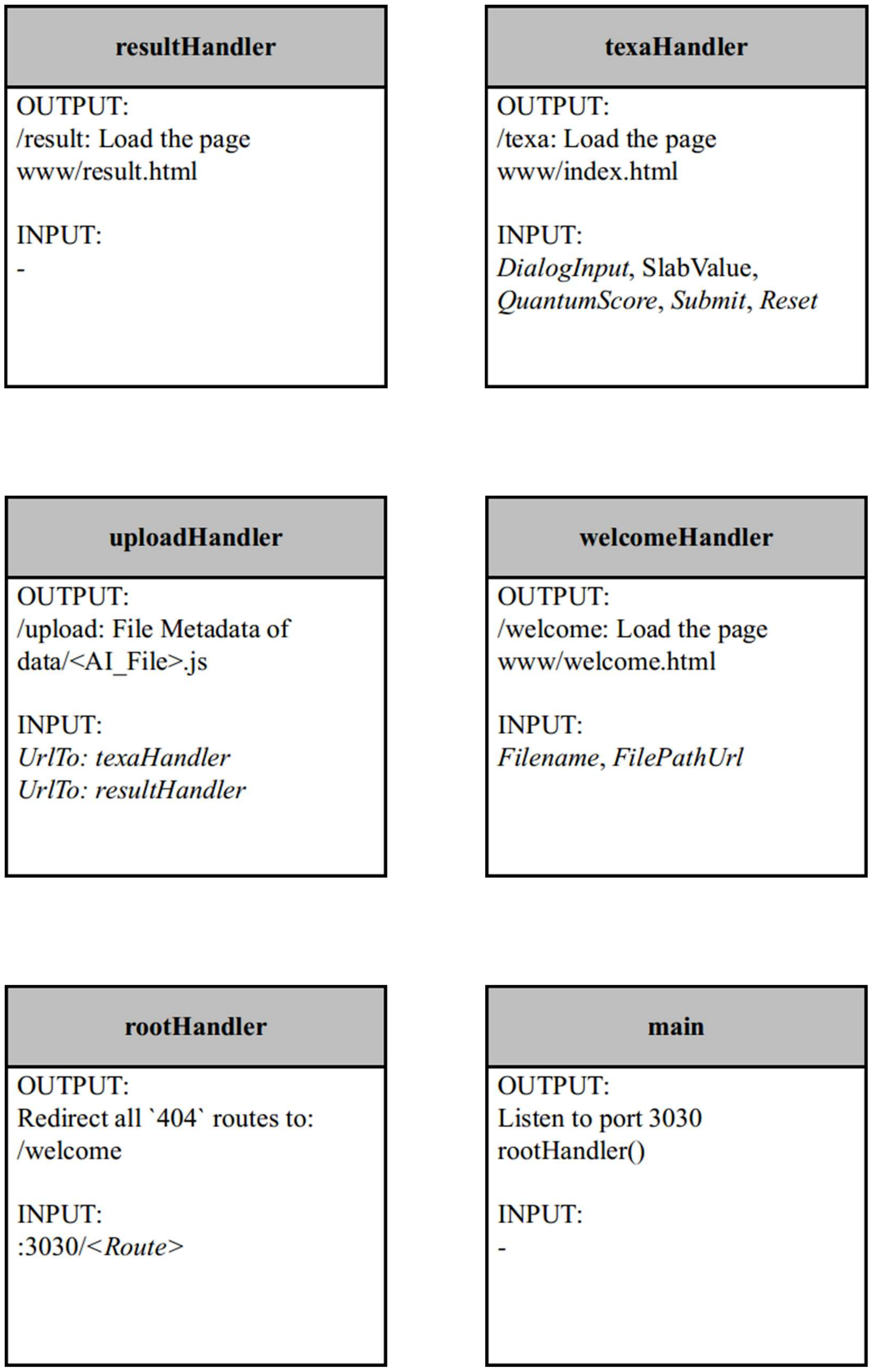

resultHandler
OUTPUT:
/result: Load the page www/result.html
INPUT:
-
texaHandler
OUTPUT:
/texa: Load the page www/index.html
INPUT:
DialogInput, SlabValue, *QuantumScore*, *Submit*, *Reset*
uploadHandler
OUTPUT:
/upload: File Metadata of data/<AI_File>.js
INPUT:
UrlTo: texaHandler
UrlTo: resultHandler
welcomeHandler
OUTPUT:
/welcome: Load the page www/welcome.html
INPUT:
Filename, *FilePathUrl*
rootHandler
OUTPUT:
Redirect all `404` routes to: /welcome
INPUT:
:3030/<*Route*>
main
OUTPUT:
Listen to port 3030
rootHandler()
INPUT:
-

Fig 4.3: Class Diagram of all the modules in the system

4.5 Class Relationship:

The purpose of designing class relationship models is to understand the association relations between each class. At current stage of the project, we observe the following association:

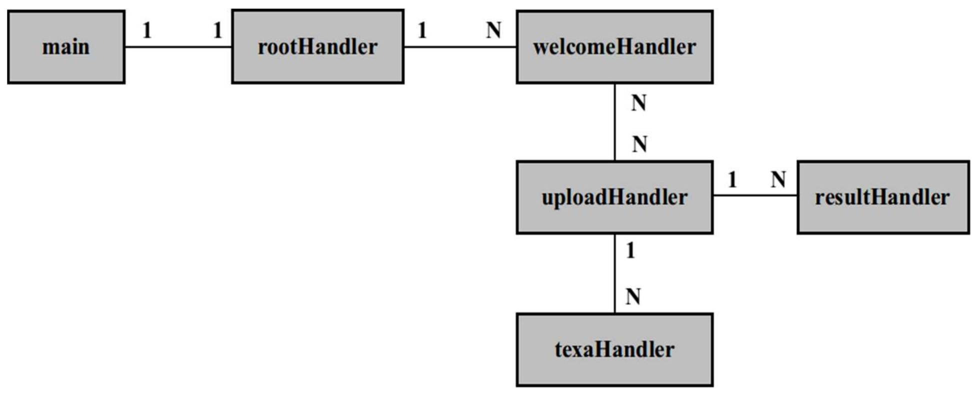
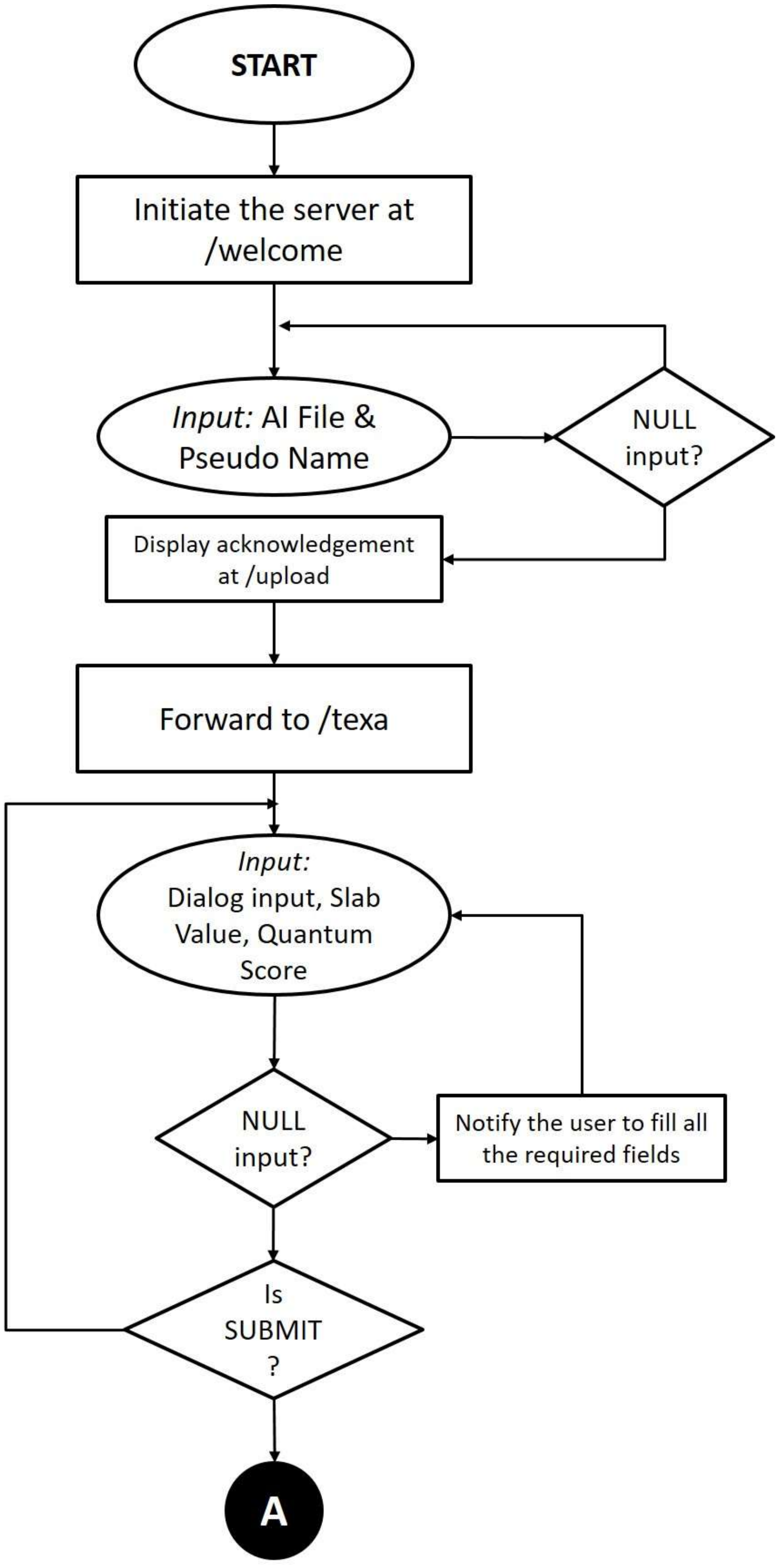


Fig 4.4: Class Relationship Diagram of all the classes in the system

4.6 System Flowchart:

The System Flowchart signifies the actual flow of the system based on the logical interaction within the system. Below mentioned is the flowchart for the proposed system:


START
Initiate the server at /welcome
Input: AI File & Pseudo Name
NULL input?
Display acknowledgement at /upload
Forward to /texa
Input: Dialog input, Slab Value, Quantum Score
NULL input?
Notify the user to fill all the required fields
Is SUBMIT ?
A

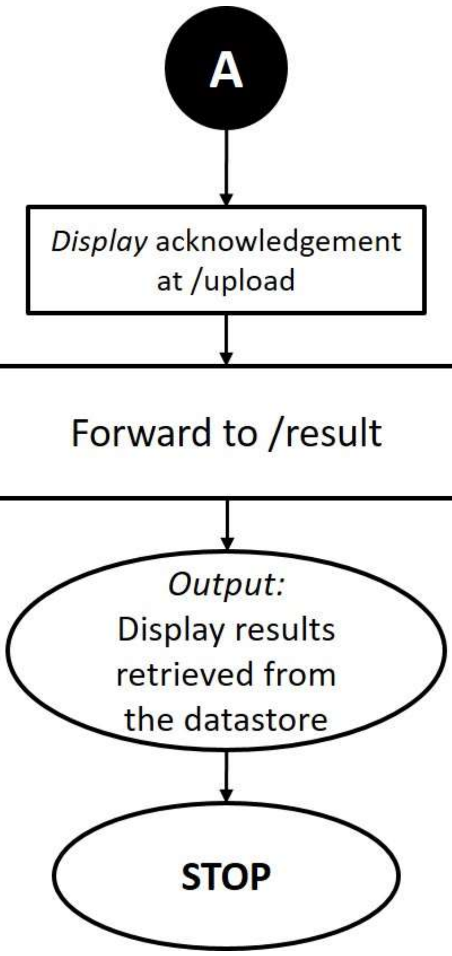

A
Display acknowledgement at /upload
Forward to /result
Output: Display results retrieved from the datastore
STOP

Fig 4.5: Flowchart diagram of the system

4.7 Graphical User Interface (GUI) Design:

The GUI is instrumental in bringing effectiveness in development and utilization of resources. All the GUI elements in this project are implemented in the form of web pages. Hence, utilizing web technologies such as XHTML, CSS & JS has been made to bring about simple, yet elegant User Experience.

4.8 The TEXA Theory:

The system is accompanied by a novel technique backed by mathematical models to provide quantitative approach under the same roof of interacting AI.

Below mentioned are the mathematical models and formulae that are used throughout the competitive assessment framework:

4.8.1 Transaction Series:

$$f(m) = t_1 + t_2 + t_3 + \ldots + t_n$$

$$f(m) = A_0 + H_1 + A_2 + H_3 + \ldots$$

$$T_n = 2(n-1) => A_{2(n-1)}$$

$$T_n = 2n-1 => H_{2n-1}$$

where, each interaction represents a bi-dialogue exchange during the Human-Machine Interaction. Each exchange of dialogue is called "Transaction" and hence, one interaction represents 2 transactions.

4.8.2 Quantum Scores:

$$A_{2(n-1)},\ H_{2n-1}\ (\forall\ n>1)$$

where, each quantum score represents whether the transaction represented by the AI is acceptable as per the Turing Test design norms.

4.8.3 Mean Test Score (MTS):

$$MTS_{AI} = \frac{\sum_{n=1}^{N} A_{2(n-1)}}{N}\ ;(\forall\ n>2)$$

$$MTS_{HI} = \frac{\sum_{n=1}^{N} H_{2n-1}}{N}\ ;(\forall\ n>2)$$

where, MTS is the average of all the quantum scores in individual test instances.

4.8.4 Slab Performance Factor (SPF):

$$SPF_i = \frac{Response\ Factor_{Slab\,i}}{Error\ Factor_{Slab\,i}}$$

$$Response\ Factor_{Slab\,i} = \frac{\sum\ No.of\ Questions\ answered\ (i.e.,\ A_{2(n-1)})}{SizeOf\,(Slab_i)}\ [\forall\ n \in SLAB_i]$$

$$Error\ Factor_{Slab\,i} = \frac{\sum\ No.of\ Questions\ failed\ overall\ (i.e.,\ A_{2(n-1)})}{Mean(SizeOf\,(Slab_i))}\ [\forall\ n \in SLAB_i]$$

$$where,\ SizeOf\,(Slab_i) = n(Slab_i);$$

$$Mean(SizeOf\,(Slab_i)) = \frac{\sum_{t=1}^{u} n(Slab_i)}{u}\ (\forall\ u > 1,\ slab\,occurences)$$

where, SPF emphasizes on the reactivity of the AI to a specific category or behaviour.

Chapter 5

IMPLEMENTATION

We now present the source code implementation of the proposed system. The system is divided into 3 major modules:

5.1 *texa:* (https://github.com/TexaProject/texa)

Hand coded 600+ lines of code for browser server engine in GoLang, JS, HTML. This module focuses on delivering a uni-platform that communicates with AI.

The following pseudo code deals with the initiation of the TEXA Server with multiple instructions to handle different requests and file organization:

```
func main() {
    fmt.Println("--TEXA SERVER--")
    fmt.Println("STATUS: INITIATED")
    fmt.Println("ADDR: http://127.0.0.1:3030")

    fsj := http.FileServer(http.Dir("www/js"))
    http.Handle("/js/", http.StripPrefix("/js/", fsj))
    fsd := http.FileServer(http.Dir("www/data"))
    http.Handle("/data/", http.StripPrefix("/data/", fsd))

    http.HandleFunc("/", rootHandler)
    http.HandleFunc("/welcome", welcomeHandler)
    http.HandleFunc("/upload", uploadHandler)
    http.HandleFunc("/texa", texaHandler)
    http.HandleFunc("/result", resultHandler)

    http.ListenAndServe(":3030", nil)
}
```

The following code manages the logic for designated routes as shown:

```
func rootHandler(w http.ResponseWriter, r *http.Request) {
    http.Redirect(w, r, "/welcome", 301)
}

func texaHandler(w http.ResponseWriter, r *http.Request) {
    fmt.Println("method:", r.Method) //get  request method
    if r.Method == "GET" {
        t, _ := template.ParseFiles("www/index.html")
        t.Execute(w, nil)
    } else {
        r.ParseForm()
        // fmt.Printf("%+v\n", r.Form)

        fmt.Println("--INTERROGATION FORM DATA--")
        IntName = r.Form.Get("IntName")
        QSA := r.Form.Get("scoreArray")
        SlabName := r.Form.Get("SlabName")
        slabSequence := r.Form.Get("slabSequence")

        fmt.Println("###", AIName)
        fmt.Println("###", IntName)
        fmt.Println("###", QSA)
        fmt.Println("###", SlabName)
        fmt.Println("###", slabSequence)

        // LOGIC
        re := regexp.MustCompile("[0-1]+")
        array := re.FindAllString(QSA, -1)

        SlabNameArray := regexp.MustCompile("[,]").Split(SlabName, -1)
        slabSeqArray := regexp.MustCompile("[,]").Split(slabSequence, -1)

        fmt.Println("###Resulting Array:")
        for x := range array {
            fmt.Println(array[x])
        }

        fmt.Println("###SlabNameArray: ")
        fmt.Println(SlabNameArray)

        fmt.Println("###slabSeqArray: ")
        fmt.Println(slabSeqArray)

        ArtiQSA := texalib.Convert(array)
        fmt.Println("###ArtiQSA:")
        fmt.Println(ArtiQSA)

        HumanQSA := texalib.SetHumanQSA(ArtiQSA)
        fmt.Println("###HumanQSA:")
        fmt.Println(HumanQSA)

        TSA := texalib.GetTransactionSeries(ArtiQSA, HumanQSA)
        fmt.Println("###TSA:")
        fmt.Println(TSA)

        ArtiMts := texalib.GetMeanTestScore(ArtiQSA)
        HumanMts := texalib.GetMeanTestScore(HumanQSA)
        fmt.Println("###ArtiMts: ", ArtiMts)
        fmt.Println("###HumanMts: ", HumanMts)

        PageArray := texajson.GetPages()
        fmt.Println("###PageArray")
        fmt.Println(PageArray)
        for _, p := range PageArray {
            fmt.Println(p)
        }

        newPage := texajson.ConvtoPage(AIName, IntName, ArtiMts, HumanMts)

        PageArray = texajson.AddtoPageArray(newPage, PageArray)
        fmt.Println("###AddedPageArray")
        fmt.Println(PageArray)

        JsonPageArray := texajson.ToJson(PageArray)
        fmt.Println("###jsonPageArray:")
        fmt.Println(JsonPageArray)

        ////
        fmt.Println("### SLAB LOGIC")

        slabPageArray := texajson.GetSlabPages()
        fmt.Println("###slabPageArray")
        fmt.Println(slabPageArray)

        slabPages := texajson.ConvtoSlabPage(ArtiQSA, SlabNameArray,
slabSeqArray)
        fmt.Println("###slabPages")
        fmt.Println(slabPages)
        for z := 0; z < len(slabPages); z++ {
            slabPageArray = texajson.AddtoSlabPageArray(slabPages[z],
slabPageArray)
        }
        fmt.Println("###finalslabPageArray")
        fmt.Println(slabPageArray)

        JsonSlabPageArray := texajson.SlabToJson(slabPageArray)
        fmt.Println("###JsonSlabPageArray: ")
        fmt.Println(JsonSlabPageArray)

        ////
        fmt.Println("### CAT LOGIC")

        CatPageArray := texajson.GetCatPages()
        fmt.Println("###CatPageArray")
        fmt.Println(CatPageArray)
        CatPages := texajson.ConvtoCatPage(AIName, slabPageArray,
SlabNameArray)
        fmt.Println("###CatPages")
        fmt.Println(CatPages)
        CatPageArray = texajson.AddtoCatPageArray(CatPages, CatPageArray)

        // for z := 0; z < len(CatPages); z++ {
        //  CatPageArray = texajson.AddtoCatPageArray(CatPages[z],
CatPageArray)
        // }
        fmt.Println("###finalCatPageArray")
        fmt.Println(CatPageArray)

        JsonCatPageArray := texajson.CatToJson(CatPageArray)
        fmt.Println("###JsonCatPageArray: ")
        fmt.Println(JsonCatPageArray)

        if r.Method == "POST" {
            http.Redirect(w, r, "/result", http.StatusSeeOther)
        }
    }
}

func welcomeHandler(w http.ResponseWriter, r *http.Request) {
    fmt.Println("method:", r.Method) //get  request method
    if r.Method == "GET" {
        t, _ := template.ParseFiles("www/welcome.html")
        t.Execute(w, nil)
    } else {
        r.ParseForm()
    }
}

// upload logic
func uploadHandler(w http.ResponseWriter, r *http.Request) {
    fmt.Println("method:", r.Method)
    if r.Method == "GET" {
        crutime := time.Now().Unix()
        h := md5.New()
        io.WriteString(h, strconv.FormatInt(crutime, 10))
        token := fmt.Sprintf("%x", h.Sum(nil))

        t, _ := template.ParseFiles("login.html")
        t.Execute(w, token)
    } else {
        r.ParseMultipartForm(32 << 20)
        file, handler, err := r.FormFile("uploadfile")
        if err != nil {
            fmt.Println(err)
            return
        }
        AIName = r.FormValue("AIName")
        fmt.Println(AIName)
        defer file.Close()

        fmt.Fprint(w, "ACKNOWLEDGEMENT:\nUploaded the file. Header
Info:\n")
        fmt.Fprintf(w, "%v", handler.Header)
        fmt.Fprint(w, "\n\nVISIT: /texa for interrogation.")
        f, err := os.OpenFile("./www/js/"+handler.Filename,
os.O_WRONLY|os.O_CREATE, 0666)
        if err != nil {
            fmt.Println(err)
            return
        }
        fmt.Println("Selected file: ", handler.Filename)
        defer f.Close()
        io.Copy(f, file)
        // http.Redirect(w, r, "/texa", 301)
    }
}

func resultHandler(w http.ResponseWriter, r *http.Request) {
    fmt.Println("method:", r.Method) //get  request method
    if r.Method == "GET" {
        t, _ := template.ParseFiles("www/result.html")
        t.Execute(w, nil)
    } else {
        r.ParseForm()
    }
}
```

5.2 ***texalib:*** (https://github.com/TexaProject/texalib)

Hand coded 50+ lines of readable code for a simplified quantification library in GoLang. This module provides mathematical functionalities required to make assessment of AIs.

The following pseudo code provides the basic mathematical functionalities required to assess the AI:

```
func Convert(xs []string) []uint64 {
    ArtiQSA := make([]uint64, 0, len(xs))

    for x := range xs {
        if s, err := strconv.ParseUint(xs[x], 10, 64); err == nil {
            // fmt.Printf("%T, %v\n", s, s)
            ArtiQSA = append(ArtiQSA, s)
        }
    }
    // fmt.Printf("%T %v", ArtiQSA, ArtiQSA)
    return ArtiQSA
}

// Total calculates the sum total of the given array
func Total(xs []uint64) uint64 {
    var total uint64
    total = 0
    for _, x := range xs {
        total += x
    }
    return total
}

// Average calculates the average of the given array
func Average(xs []uint64) float64 {
    return float64(float64(Total(xs)) / float64(uint64(len(xs))))
}

// SetHumanQSA creates and returns the Quantum Score Array for Human
Intelligence
func SetHumanQSA(xs []uint64) []uint64 {
    HumanQSA := make([]uint64, 0, len(xs))
    for _, x := range xs {
        if x == 1 {
            HumanQSA = append(HumanQSA, 0)
        }
        if x == 0 {
            HumanQSA = append(HumanQSA, 1)
        }
    }
    return HumanQSA
}

// GetTransactionSeries return an array of alternate QSA values
func GetTransactionSeries(ArtiQSA []uint64, HumanQSA []uint64) []uint64 {
    TSA := make([]uint64, 0, len(ArtiQSA)+len(HumanQSA))
    for index := 0; index < len(ArtiQSA); index++ {
        TSA = append(TSA, ArtiQSA[index])
        TSA = append(TSA, HumanQSA[index])
    }
    return TSA
}

// GetMeanTestScore returns the average of the QSA of AI & HI - ArtiQSA &
HumanQSA respectively
```

```
func GetMeanTestScore(QSA []uint64) float64 {
    if len(QSA) <= 2 {
        fmt.Println("ERROR: Min. of 2 transactions required to compute
M.T.S.")
        panic("ERROR: QSA_LEN_ERR")
    }
    return (Average(QSA))
}
```

5.3 *texajson:* (https://github.com/TexaProject/texajson)

Hand coded 275+ lines of code for a robust JSON library in GoLang. This module serves as a native Datastore interpreter.

A simple pseudo code with structures for parsing the MTS Matrix:

```
// Page exports schema for mts.json
type Page struct {
    AIName   string  `json:"AIName"`
    IntName  string  `json:"IntName"`
    ArtiMts  float64 `json:"ArtiMts"`
    HumanMts float64 `json:"HumanMts"`
}

// ToString returns the string equivalent JSON format of Page
func (p Page) ToString() string {
    return ToJson(p)
}

//GetPages returns a converted Page Array persistent to the mts.json
func GetPages() []Page {
    raw, err := ioutil.ReadFile("./www/data/mts.json")
    if err != nil {
        fmt.Println(err.Error())
        os.Exit(1)
    }

    var c []Page
    json.Unmarshal(raw, &c)
    return c
}

// ConvtoPage converts a set of data vars into a Page struct variable
func ConvtoPage(AIName string, IntName string, ArtiMts float64, HumanMts
float64) Page {
    var newPage Page
    newPage.AIName = AIName
    newPage.IntName = IntName
    newPage.ArtiMts = ArtiMts
    newPage.HumanMts = HumanMts
    return newPage
}

// AddtoPageArray Appends a new page 'p' to the specified target PageArray
'pa'
func AddtoPageArray(p Page, pa []Page) []Page {
    for x := range pa {
        if (p.AIName == pa[x].AIName) && (p.IntName == pa[x].IntName) {
            pa[x].ArtiMts = p.ArtiMts
            pa[x].HumanMts = p.HumanMts
            // panic("JSON ERROR: Can't append a Duplicate Page into
PageArray")
            return pa
        }
    }
    return (append(pa, p))
}

// ToJson marshals PageArray data into JSON format
func ToJson(p interface{}) string {
    bytes, err := json.Marshal(p)
    if err != nil {
        fmt.Println(err.Error())
        os.Exit(1)
    }
    ioutil.WriteFile("./www/data/mts.json", bytes, 0644)
    return string(bytes)
}
```

Similar pseudo code, yet with an additional SPF calculation logic for parsing and building CAT Matrix:

```
type CatValArray struct {
    CatName string  `json:"CatName"`
    Spf     float64 `json:"Spf"`
}

// CatPage exports schema for data/cat.json
type CatPage struct {
    AIName string         `json:"AIName"`
    CatVal []CatValArray `json:"CatVal"`
}

// ToString returns the string equivalent JSON format of CatPage
```

```go
func (p CatPage) ToString() string {
    return ToJson(p)
}

//GetCatPages returns a converted CatPage Array persistent to the mts.json
func GetCatPages() []CatPage {
    raw, err := ioutil.ReadFile("./www/data/cat.json")
    if err != nil {
        fmt.Println(err.Error())
        os.Exit(1)
    }

    var c []CatPage
    json.Unmarshal(raw, &c)
    return c
}

// ConvtoCatPage converts a set of data vars into a CatPage struct variable
func ConvtoCatPage(AIName string, slabPageArray []SlabPage, SlabNameArray
[]string) CatPage {
    var newCatPage CatPage
    newCatPage.AIName = AIName
    cv := make([]CatValArray, len(SlabNameArray))

    fmt.Println("####SlabTempSize")
    fmt.Println(SlabTempSize)

    for index := 0; index < len(slabPageArray); index++ {
        for n := 0; n < len(SlabNameArray); n++ {
            if SlabNameArray[n] == slabPageArray[index].SlabName {
                cv[n].CatName = slabPageArray[index].SlabName
                ef := (float64(slabPageArray[index].NQDropped) /
float64(slabPageArray[index].AvgSlabSize))
                rf := (float64(SlabTempSize[n]-SlabTempNQD[n]) /
float64(SlabTempSize[n]))
                // cv[index].Spf = ((float64(SlabTempSize[n]-
slabPageArray[index].NQDropped) / float64(SlabTempSize[n])) /
(float64(slabPageArray[index].NQDropped) /
float64(slabPageArray[index].AvgSlabSize)))
                spfTemp := rf / ef
                if math.IsInf(spfTemp, 0) {
                    cv[n].Spf = 999
                } else {
                    cv[n].Spf = spfTemp
                }
            }
        }
    }
    fmt.Println("####cv")
    fmt.Println(cv)

    newCatPage.CatVal = cv
    return newCatPage
}

// AddtoCatPageArray Appends a new CatPage 'p' to the specified target
CatPageArray 'pa'
func AddtoCatPageArray(p CatPage, pa []CatPage) []CatPage {
    for index := 0; index < len(pa); index++ {
        if pa[index].AIName == p.AIName {
            for a := 0; a < len(p.CatVal); a++ {
                for m := 0; m < len(pa[index].CatVal); m++ {
                    if p.CatVal[a].CatName == pa[index].CatVal[m].CatName {
                        pa[index].CatVal[m].Spf = p.CatVal[a].Spf
                    }
                }
                pa[index].CatVal = append(pa[index].CatVal, p.CatVal[a])
                // pa[index].CatVal = append(pa[index].CatVal, p.CatVal[a])
            }
            return pa
        }
    }
    return (append(pa, p))
}

// ToJson marshals CatPageArray data into JSON format
func CatToJson(p interface{}) string {
    bytes, err := json.Marshal(p)
    if err != nil {
        fmt.Println(err.Error())
        os.Exit(1)
    }
    ioutil.WriteFile("./www/data/cat.json", bytes, 0644)
    SlabTempSize = nil
    SlabTempNQD = nil
    return string(bytes)
}

// SlabPage exports schema for reportcard/mts.json
type SlabPage struct {
    SlabName     string `json:"SlabName"`
    NQDropped    int    `json:"NQDropped"`
    AvgSlabSize  int    `json:"AvgSlabSize"`
    NSlabExposed int    `json:"NSlabExposed"`
}

// ToString returns the string equivalent JSON format of SlabPage
```

```
func (p SlabPage) ToString() string {
    fmt.Println("###SlabToString()")
    return ToJson(p)
}

//GetSlabPages returns a converted SlabPage Array persistent to the
mts.json
func GetSlabPages() []SlabPage {
    fmt.Println("###GetSlabPages()")
    raw, err := ioutil.ReadFile("./www/data/reportcard/slab.json")
    if err != nil {
        fmt.Println(err.Error())
        os.Exit(1)
    }

    var c []SlabPage
    json.Unmarshal(raw, &c)
    return c
}

func dupCount(list []string) map[string]int {
    fmt.Println("###dupCount()")
    duplicate_frequency := make(map[string]int)

    for _, item := range list {
        // check if the item/element exist in the duplicate_frequency map

        _, exist := duplicate_frequency[item]

        if exist {
            duplicate_frequency[item] += 1 // increase counter by 1 if
already in the map
        } else {
            duplicate_frequency[item] = 1 // else start counting from 1
        }
    }
    return duplicate_frequency
}

// ConvtoSlabPage configures the parameters of SlabPage using the ArtiQSA
func ConvtoSlabPage(ArtiQSA []uint64, SlabNameArray []string, slabSeqArray
[]string) []SlabPage {
    fmt.Println("###ConvtoSlabPage()")
    sp := make([]SlabPage, len(SlabNameArray))

    // Initialization for all Slabs mentioned in the SlabNameArray
    for k := 0; k < len(SlabNameArray); k++ {
        sp[k].SlabName = SlabNameArray[k]
        sp[k].NQDropped = 0
        sp[k].AvgSlabSize = 0
        sp[k].NSlabExposed = 0
    }
    fmt.Println("###sp")
    fmt.Println(sp)

    SlabTempNQD = make([]int, len(SlabNameArray))
    for i := 0; i < len(ArtiQSA); i++ {
        if ArtiQSA[i] == 0 {
            for k := 0; k < len(SlabNameArray); k++ {
                if sp[k].SlabName == slabSeqArray[i] {
                    SlabTempNQD[k]++
                    sp[k].NQDropped++
                }
            }
        }
    }
    fmt.Println("###sp-postNQDropped")
    fmt.Println(sp)

    fmt.Println("###dupMap")
    dupMap := dupCount(slabSeqArray)
    SlabTempSize = make([]int, len(SlabNameArray))
    for k, v := range dupMap {
        for x := 0; x < len(SlabNameArray); x++ {
            if k == SlabNameArray[x] {
                if v >= 1 {
                    SlabTempSize[x] = v
                    sp[x].AvgSlabSize =
(sp[x].AvgSlabSize*sp[x].NSlabExposed + v) / (sp[x].NSlabExposed + 1)
                    sp[x].NSlabExposed++
                }
            }
        }
    }
    return (sp)
}

// AddtoSlabPageArray Appends a new Slabpage 'p' to the specified target
SlabPageArray 'pa'
func AddtoSlabPageArray(p SlabPage, pa []SlabPage) []SlabPage {
    fmt.Println("###AddtoSlabPageArray()")
    for x := 0; x < len(pa); x++ {
        if p.SlabName == pa[x].SlabName {
            // panic("JSON ERROR: Can't append a Duplicate SlabPage into
SlabPageArray")
            pa[x].NSlabExposed += p.NSlabExposed
            pa[x].NQDropped += p.NQDropped
            pa[x].AvgSlabSize = (pa[x].AvgSlabSize + p.AvgSlabSize) /
pa[x].NSlabExposed
            return pa
        }

    }
    return (append(pa, p))
}

// SlabToJson marshals SlabPageArray data into JSON format
func SlabToJson(p interface{}) string {
    fmt.Println("###SlabToJson()")
    bytes, err := json.Marshal(p)
    if err != nil {
        fmt.Println(err.Error())
        os.Exit(1)
    }
    ioutil.WriteFile("./www/data/reportcard/slab.json", bytes, 0644)
    return string(bytes)
}
```

5.4 Native Go libraries:

We have also leveraged several native Go Libraries in order to provide robust performance & experience qualities.

A typical pseudo code importing all the libraries mentioned below:

```
import (
    "crypto/md5"

    "github.com/TexaProject/texajson"
    "github.com/TexaProject/texalib"
    "fmt"
    "html/template"
    "io"
    "net/http"
    "os"
    "regexp"
    "strconv"
    "time"
)
```

5.4.1 net/http:

Native support for several networking requirements including client-server functionalities.

5.4.2 *crypto/md5:*

Native support for cryptographic requirements. Utilized in the project only for generating the Hash/Checksum for the uploaded AI files.

5.4.3 *html/template:*

Native support for dynamic web component generation and data binding requirements. Utilized in the project for generating acknowledgement webpage & so on.

Chapter 6

TESTING

6.1 Testing GoLang Server Program:

Testing the functionality of programs in Go Languages is a automatically generated, and swift.

The following test code generated tests the main.go of the texa:

```go
package main

import (
    "net/http"
    "testing"
)

func Test_rootHandler(t *testing.T) {
    type args struct {
        w http.ResponseWriter
        r *http.Request
    }
    tests := []struct {
        name string
        args args
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            rootHandler(tt.args.w, tt.args.r)
        })
    }
}

func Test_texaHandler(t *testing.T) {
    type args struct {
        w http.ResponseWriter
        r *http.Request
    }
    tests := []struct {
        name string
        args args
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            texaHandler(tt.args.w, tt.args.r)
        })
    }
}

func Test_welcomeHandler(t *testing.T) {
    type args struct {
        w http.ResponseWriter
        r *http.Request
    }
    tests := []struct {
        name string
        args args
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            welcomeHandler(tt.args.w, tt.args.r)
        })
    }
}

func Test_uploadHandler(t *testing.T) {
    type args struct {
        w http.ResponseWriter
        r *http.Request
    }
    tests := []struct {
        name string
        args args
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            uploadHandler(tt.args.w, tt.args.r)
        })
    }
}

func Test_resultHandler(t *testing.T) {
    type args struct {
        w http.ResponseWriter
        r *http.Request
    }
    tests := []struct {
        name string
        args args
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            resultHandler(tt.args.w, tt.args.r)
        })
    }
}

func Test_main(t *testing.T) {
    tests := []struct {
        name string
    }{
    // TODO: Add test cases.
    }
    for range tests {
        t.Run(tt.name, func(t *testing.T) {
            main()
        })
    }
}
```

6.2 Testing GoLang libraries:

Testing the functionality of libraries or packages are difficult in other programming landscapes. However, this doesn't apply to the case of Go Programming Language and hence the choice of implementation.

The following generated test code tests for texalib package's functionalities:

```
func TestConvert(t *testing.T) {
    type args struct {
        xs []string
    }
    tests := []struct {
        name string
        args args
        want []uint64
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := Convert(tt.args.xs); !reflect.DeepEqual(got, tt.want)
{
                t.Errorf("Convert() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestTotal(t *testing.T) {
    type args struct {
        xs []uint64
    }
    tests := []struct {
        name string
        args args
        want uint64
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := Total(tt.args.xs); got != tt.want {
                t.Errorf("Total() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestAverage(t *testing.T) {
    type args struct {
        xs []uint64
    }
    tests := []struct {
        name string
        args args
        want float64
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := Average(tt.args.xs); got != tt.want {
                t.Errorf("Average() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestSetHumanQSA(t *testing.T) {
    type args struct {
        xs []uint64
    }
    tests := []struct {
        name string
        args args
        want []uint64
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := SetHumanQSA(tt.args.xs); !reflect.DeepEqual(got,
tt.want) {
                t.Errorf("SetHumanQSA() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestGetTransactionSeries(t *testing.T) {
    type args struct {
        ArtiQSA  []uint64
        HumanQSA []uint64
    }
    tests := []struct {
        name string
        args args
        want []uint64
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := GetTransactionSeries(tt.args.ArtiQSA,
tt.args.HumanQSA); !reflect.DeepEqual(got, tt.want) {
                t.Errorf("GetTransactionSeries() = %v, want %v", got,
tt.want)
            }
        })
    }
}
```

```go
func TestGetMeanTestScore(t *testing.T) {
    type args struct {
        QSA []uint64
    }
    tests := []struct {
        name string
        args args
        want float64
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := GetMeanTestScore(tt.args.QSA); got != tt.want {
                t.Errorf("GetMeanTestScore() = %v, want %v", got, tt.want)
            }
        })
    }
}
```

The following generated test code tests for texajson package’s functionalities:

```go
func TestCatPage_ToString(t *testing.T) {
    tests := []struct {
        name string
        p    CatPage
        want string
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := tt.p.ToString(); got != tt.want {
                t.Errorf("CatPage.ToString() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestGetCatPages(t *testing.T) {
    tests := []struct {
        name string
        want []CatPage
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := GetCatPages(); !reflect.DeepEqual(got, tt.want) {
                t.Errorf("GetCatPages() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestConvtoCatPage(t *testing.T) {
    type args struct {
        AIName        string
        slabPageArray []SlabPage
        SlabNameArray []string
    }
    tests := []struct {
        name string
        args args
        want CatPage
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := ConvtoCatPage(tt.args.AIName, tt.args.slabPageArray,
tt.args.SlabNameArray); !reflect.DeepEqual(got, tt.want) {
                t.Errorf("ConvtoCatPage() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestAddtoCatPageArray(t *testing.T) {
    type args struct {
        p  CatPage
        pa []CatPage
    }
    tests := []struct {
        name string
        args args
        want []CatPage
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := AddtoCatPageArray(tt.args.p, tt.args.pa);
!reflect.DeepEqual(got, tt.want) {
                t.Errorf("AddtoCatPageArray() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestCatToJson(t *testing.T) {
    type args struct {
        p interface{}
    }
    tests := []struct {
        name string
        args args
        want string
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := CatToJson(tt.args.p); got != tt.want {
                t.Errorf("CatToJson() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestSlabPage_ToString(t *testing.T) {
    tests := []struct {
        name string
        p    SlabPage
        want string
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := tt.p.ToString(); got != tt.want {
                t.Errorf("SlabPage.ToString() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestGetSlabPages(t *testing.T) {
    tests := []struct {
        name string
        want []SlabPage
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := GetSlabPages(); !reflect.DeepEqual(got, tt.want) {
                t.Errorf("GetSlabPages() = %v, want %v", got, tt.want)
            }
        })
    }
}

func Test_dupCount(t *testing.T) {
    type args struct {
        list []string
    }
    tests := []struct {
        name string
        args args
        want map[string]int
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := dupCount(tt.args.list); !reflect.DeepEqual(got,
tt.want) {
                t.Errorf("dupCount() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestConvtoSlabPage(t *testing.T) {
    type args struct {
        ArtiQSA       []uint64
        SlabNameArray []string
        slabSeqArray  []string
    }
    tests := []struct {
        name string
        args args
        want []SlabPage
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := ConvtoSlabPage(tt.args.ArtiQSA,
tt.args.SlabNameArray, tt.args.slabSeqArray); !reflect.DeepEqual(got,
tt.want) {
                t.Errorf("ConvtoSlabPage() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestAddtoSlabPageArray(t *testing.T) {
    type args struct {
        p  SlabPage
        pa []SlabPage
    }
    tests := []struct {
        name string
        args args
        want []SlabPage
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := AddtoSlabPageArray(tt.args.p, tt.args.pa);
!reflect.DeepEqual(got, tt.want) {
                t.Errorf("AddtoSlabPageArray() = %v, want %v", got,
tt.want)
            }
        })
    }
}

func TestSlabToJson(t *testing.T) {
    type args struct {
        p interface{}
    }
    tests := []struct {
        name string
        args args
        want string
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := SlabToJson(tt.args.p); got != tt.want {
                t.Errorf("SlabToJson() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestPage_ToString(t *testing.T) {
    tests := []struct {
        name string
        p    Page
        want string
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := tt.p.ToString(); got != tt.want {
                t.Errorf("Page.ToString() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestGetPages(t *testing.T) {
    tests := []struct {
        name string
        want []Page
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := GetPages(); !reflect.DeepEqual(got, tt.want) {
                t.Errorf("GetPages() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestConvtoPage(t *testing.T) {
    type args struct {
        AIName   string
        IntName  string
        ArtiMts  float64
        HumanMts float64
    }
    tests := []struct {
        name string
        args args
        want Page
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := ConvtoPage(tt.args.AIName, tt.args.IntName,
tt.args.ArtiMts, tt.args.HumanMts); !reflect.DeepEqual(got, tt.want) {
                t.Errorf("ConvtoPage() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestAddtoPageArray(t *testing.T) {
    type args struct {
        p  Page
        pa []Page
    }
    tests := []struct {
        name string
        args args
        want []Page
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := AddtoPageArray(tt.args.p, tt.args.pa);
!reflect.DeepEqual(got, tt.want) {
                t.Errorf("AddtoPageArray() = %v, want %v", got, tt.want)
            }
        })
    }
}

func TestToJson(t *testing.T) {
    type args struct {
        p interface{}
    }
    tests := []struct {
        name string
        args args
        want string
    }{
    // TODO: Add test cases.
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            if got := ToJson(tt.args.p); got != tt.want {
                t.Errorf("ToJson() = %v, want %v", got, tt.want)
            }
        })
    }
}
```

6.3 Integration & System testing:

The integration and system testing is taken care by the go testing tool chain usually. However, since our system doesn’t require integration with external systems, the existing test scripts suffice the test adoption requirements.

Chapter 7

SNAPSHOTS AND RESULTS

7.1 Input Specification:

The input data in the proposed system has a set of conditions as follows:

1. All input variables can't be empty
2. All input variables can't be out of range
3. All input variables can't be negative
4. AI's Name & Interrogator's Name can be alphanumeric array only
5. AI file to be uploaded must be in the JavaScript implementation format of ELIZA

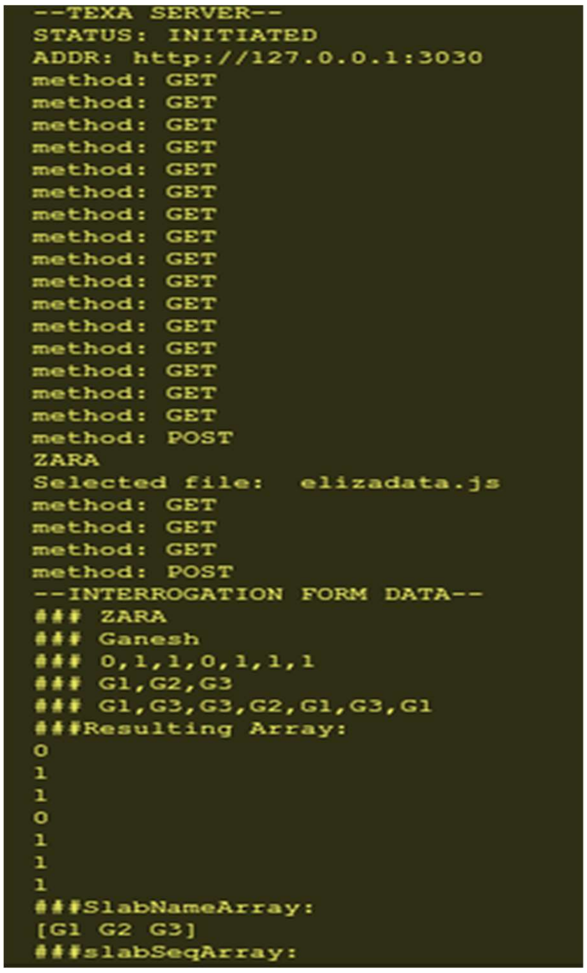
7.2 Output Specification:

The output in the system has a few set of characteristics:

1. Session data during interrogation is displayed through the JS Alert feature
2. Entire session data is dumped as an output once the session is submitted for calculation in the form of console output.

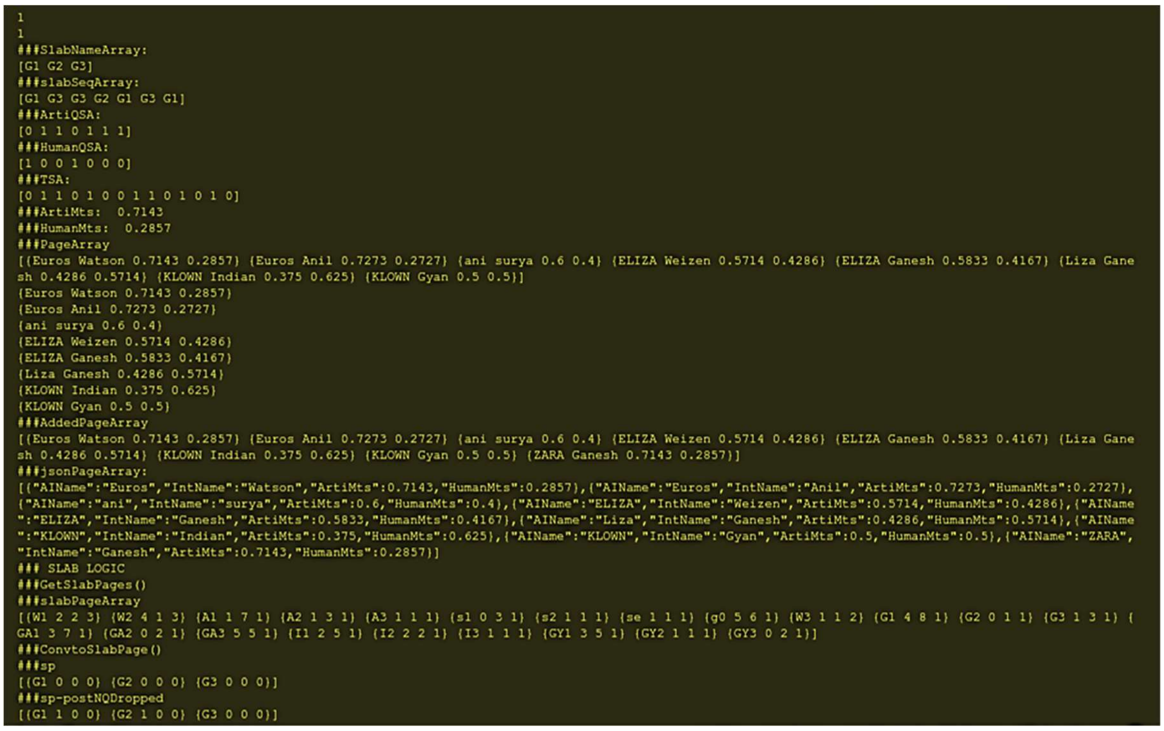
7.3 Result Snapshots:

The console output on operating the server is as obtained below:



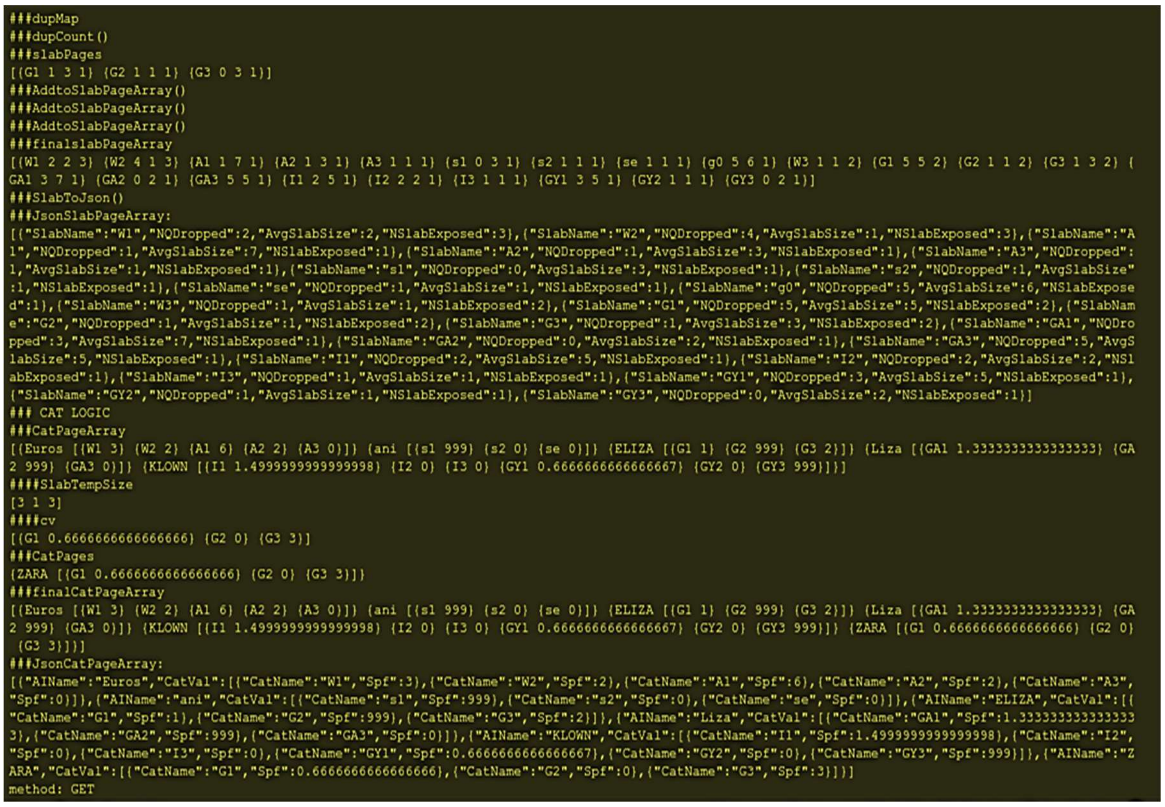
```
--TEXA SERVER--
STATUS: INITIATED
ADDR: http://127.0.0.1:3030
method: GET
method: GET
method: GET
method: GET
method: GET
method: GET
method: GET
method: GET
method: GET
method: GET
method: GET
method: GET
method: GET
method: GET
method: GET
method: GET
method: POST
ZARA
Selected file:  elizadata.js
method: GET
method: GET
method: GET
method: POST
--INTERROGATION FORM DATA--
### ZARA
### Ganesh
### 0,1,1,0,1,1,1
### G1,G2,G3
### G1,G3,G3,G2,G1,G3,G1
###Resulting Array:
0
1
1
0
1
1
1
###SlabNameArray:
[G1 G2 G3]
###slabSeqArray:
```

Fig 7.1: Console output of TEXA Server



```
1
1
###SlabNameArray:
[G1 G2 G3]
###slabSeqArray:
[G1 G3 G3 G2 G1 G3 G1]
###ArtiQSA:
[0 1 1 0 1 1 1]
###HumanQSA:
[1 0 0 1 0 0 0]
###TSA:
[0 1 1 0 1 0 0 1 1 0 1 0 1 0]
###ArtiMts:  0.7143
###HumanMts:  0.2857
###PageArray
[{Euros Watson 0.7143 0.2857} {Euros Anil 0.7273 0.2727} {ani surya 0.6 0.4} {ELIZA Weizen 0.5714 0.4286} {ELIZA Ganesh 0.5833 0.4167} {Liza Gane
sh 0.4286 0.5714} {KLOWN Indian 0.375 0.625} {KLOWN Gyan 0.5 0.5}]
{Euros Watson 0.7143 0.2857}
{Euros Anil 0.7273 0.2727}
{ani surya 0.6 0.4}
{ELIZA Weizen 0.5714 0.4286}
{ELIZA Ganesh 0.5833 0.4167}
{Liza Ganesh 0.4286 0.5714}
{KLOWN Indian 0.375 0.625}
{KLOWN Gyan 0.5 0.5}
###AddedPageArray
[{Euros Watson 0.7143 0.2857} {Euros Anil 0.7273 0.2727} {ani surya 0.6 0.4} {ELIZA Weizen 0.5714 0.4286} {ELIZA Ganesh 0.5833 0.4167} {Liza Gane
sh 0.4286 0.5714} {KLOWN Indian 0.375 0.625} {KLOWN Gyan 0.5 0.5} {ZARA Ganesh 0.7143 0.2857}]
###jsonPageArray:
[{"AIName":"Euros","IntName":"Watson","ArtiMts":0.7143,"HumanMts":0.2857},{"AIName":"Euros","IntName":"Anil","ArtiMts":0.7273,"HumanMts":0.2727},
{"AIName":"ani","IntName":"surya","ArtiMts":0.6,"HumanMts":0.4},{"AIName":"ELIZA","IntName":"Weizen","ArtiMts":0.5714,"HumanMts":0.4286},{"AIName
":"ELIZA","IntName":"Ganesh","ArtiMts":0.5833,"HumanMts":0.4167},{"AIName":"Liza","IntName":"Ganesh","ArtiMts":0.4286,"HumanMts":0.5714},{"AIName
":"KLOWN","IntName":"Indian","ArtiMts":0.375,"HumanMts":0.625},{"AIName":"KLOWN","IntName":"Gyan","ArtiMts":0.5,"HumanMts":0.5},{"AIName":"ZARA",
"IntName":"Ganesh","ArtiMts":0.7143,"HumanMts":0.2857}]
### SLAB LOGIC
###GetSlabPages()
###slabPageArray
[{W1 2 2 3} {W2 4 1 3} {A1 1 7 1} {A2 1 3 1} {A3 1 1 1} {s1 0 3 1} {s2 1 1 1} {se 1 1 1} {g0 5 6 1} {W3 1 1 2} {G1 4 8 1} {G2 0 1 1} {G3 1 3 1} {
GA1 3 7 1} {GA2 0 2 1} {GA3 5 5 1} {I1 2 5 1} {I2 2 2 1} {I3 1 1 1} {GY1 3 5 1} {GY2 1 1 1} {GY3 0 2 1}]
###ConvtoSlabPage()
###sp
[{G1 0 0 0} {G2 0 0 0} {G3 0 0 0}]
###sp-postNQDropped
[{G1 1 0 0} {G2 1 0 0} {G3 0 0 0}]
```

Fig 7.2: Console output of TEXA Server with Session Data Dump



```
###dupMap
###dupCount()
###slabPages
[{G1 1 3 1} {G2 1 1 1} {G3 0 3 1}]
###AddtoSlabPageArray()
###AddtoSlabPageArray()
###AddtoSlabPageArray()
###finalslabPageArray
[{W1 2 2 3} {W2 4 1 3} {A1 1 7 1} {A2 1 3 1} {A3 1 1 1} {s1 0 3 1} {s2 1 1 1} {se 1 1 1} {g0 5 6 1} {W3 1 1 2} {G1 5 5 2} {G2 1 1 2} {G3 1 3 2} {
GA1 3 7 1} {GA2 0 2 1} {GA3 5 5 1} {I1 2 5 1} {I2 2 2 1} {I3 1 1 1} {GY1 3 5 1} {GY2 1 1 1} {GY3 0 2 1}]
###SlabToJson()
###JsonSlabPageArray:
[{"SlabName":"W1","NQDropped":2,"AvgSlabSize":2,"NSlabExposed":3},{"SlabName":"W2","NQDropped":4,"AvgSlabSize":1,"NSlabExposed":3},{"SlabName":"A
1","NQDropped":1,"AvgSlabSize":7,"NSlabExposed":1},{"SlabName":"A2","NQDropped":1,"AvgSlabSize":3,"NSlabExposed":1},{"SlabName":"A3","NQDropped":
1,"AvgSlabSize":1,"NSlabExposed":1},{"SlabName":"s1","NQDropped":0,"AvgSlabSize":3,"NSlabExposed":1},{"SlabName":"s2","NQDropped":1,"AvgSlabSize"
:1,"NSlabExposed":1},{"SlabName":"se","NQDropped":1,"AvgSlabSize":1,"NSlabExposed":1},{"SlabName":"g0","NQDropped":5,"AvgSlabSize":6,"NSlabExpose
d":1},{"SlabName":"W3","NQDropped":1,"AvgSlabSize":1,"NSlabExposed":2},{"SlabName":"G1","NQDropped":5,"AvgSlabSize":5,"NSlabExposed":2},{"SlabNam
e":"G2","NQDropped":1,"AvgSlabSize":1,"NSlabExposed":2},{"SlabName":"G3","NQDropped":1,"AvgSlabSize":3,"NSlabExposed":2},{"SlabName":"GA1","NQDro
pped":3,"AvgSlabSize":7,"NSlabExposed":1},{"SlabName":"GA2","NQDropped":0,"AvgSlabSize":2,"NSlabExposed":1},{"SlabName":"GA3","NQDropped":5,"AvgS
labSize":5,"NSlabExposed":1},{"SlabName":"I1","NQDropped":2,"AvgSlabSize":5,"NSlabExposed":1},{"SlabName":"I2","NQDropped":2,"AvgSlabSize":2,"NSl
abExposed":1},{"SlabName":"I3","NQDropped":1,"AvgSlabSize":1,"NSlabExposed":1},{"SlabName":"GY1","NQDropped":3,"AvgSlabSize":5,"NSlabExposed":1},
{"SlabName":"GY2","NQDropped":1,"AvgSlabSize":1,"NSlabExposed":1},{"SlabName":"GY3","NQDropped":0,"AvgSlabSize":2,"NSlabExposed":1}]
### CAT LOGIC
###CatPageArray
[{Euros [{W1 3} {W2 2} {A1 6} {A2 2} {A3 0}]} {ani [{s1 999} {s2 0} {se 0}]} {ELIZA [{G1 1} {G2 999} {G3 2}]} {Liza [{GA1 1.3333333333333333} {GA
2 999} {GA3 0}]} {KLOWN [{I1 1.4999999999999998} {I2 0} {I3 0} {GY1 0.6666666666666667} {GY2 0} {GY3 999}]}]
####SlabTempSize
[3 1 3]
####cv
[{G1 0.6666666666666666} {G2 0} {G3 3}]
###CatPages
{ZARA [{G1 0.6666666666666666} {G2 0} {G3 3}]}
###finalCatPageArray
[{Euros [{W1 3} {W2 2} {A1 6} {A2 2} {A3 0}]} {ani [{s1 999} {s2 0} {se 0}]} {ELIZA [{G1 1} {G2 999} {G3 2}]} {Liza [{GA1 1.3333333333333333} {GA
2 999} {GA3 0}]} {KLOWN [{I1 1.4999999999999998} {I2 0} {I3 0} {GY1 0.6666666666666667} {GY2 0} {GY3 999}]} {ZARA [{G1 0.6666666666666666} {G2 0}
 {G3 3}]}]
###JsonCatPageArray:
[{"AIName":"Euros","CatVal":[{"CatName":"W1","Spf":3},{"CatName":"W2","Spf":2},{"CatName":"A1","Spf":6},{"CatName":"A2","Spf":2},{"CatName":"A3",
"Spf":0}]},{"AIName":"ani","CatVal":[{"CatName":"s1","Spf":999},{"CatName":"s2","Spf":0},{"CatName":"se","Spf":0}]},{"AIName":"ELIZA","CatVal":[{
"CatName":"G1","Spf":1},{"CatName":"G2","Spf":999},{"CatName":"G3","Spf":2}]},{"AIName":"Liza","CatVal":[{"CatName":"GA1","Spf":1.33333333333333
3},{"CatName":"GA2","Spf":999},{"CatName":"GA3","Spf":0}]},{"AIName":"KLOWN","CatVal":[{"CatName":"I1","Spf":1.4999999999999998},{"CatName":"I2",
"Spf":0},{"CatName":"I3","Spf":0},{"CatName":"GY1","Spf":0.6666666666666667},{"CatName":"GY2","Spf":0},{"CatName":"GY3","Spf":999}]},{"AIName":"Z
ARA","CatVal":[{"CatName":"G1","Spf":0.6666666666666666},{"CatName":"G2","Spf":0},{"CatName":"G3","Spf":3}]}]
method: GET
```

Fig 7.3: Console output of TEXA Server profiling JSON Datastore

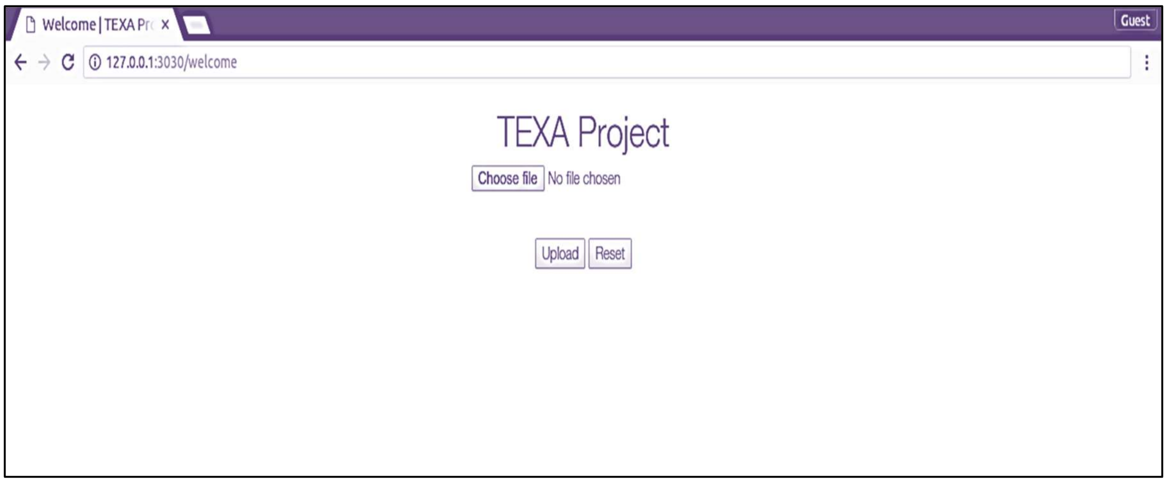


Fig 7.4: Welcome page of the system

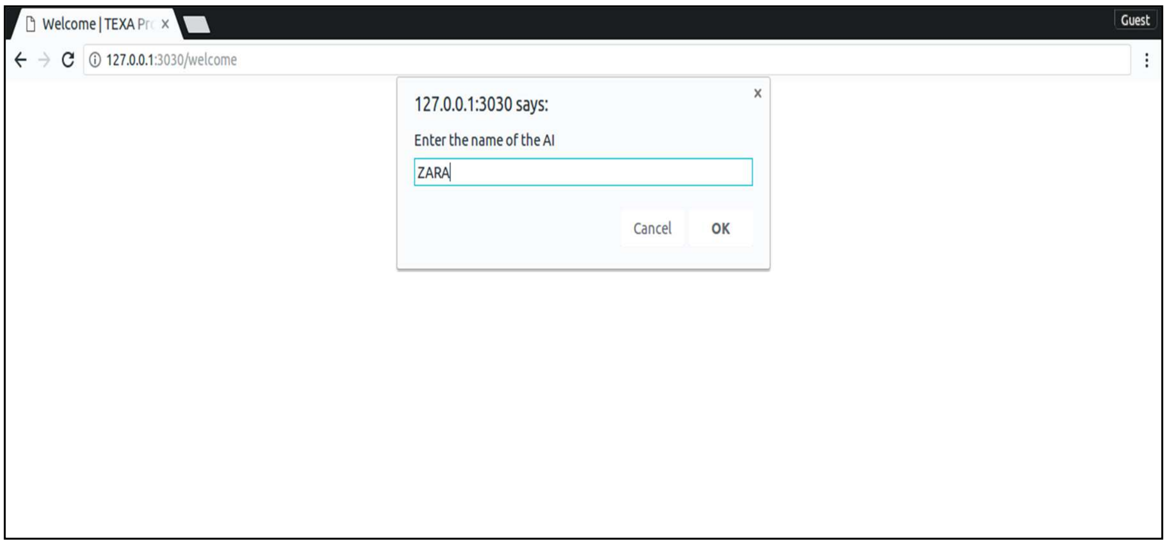
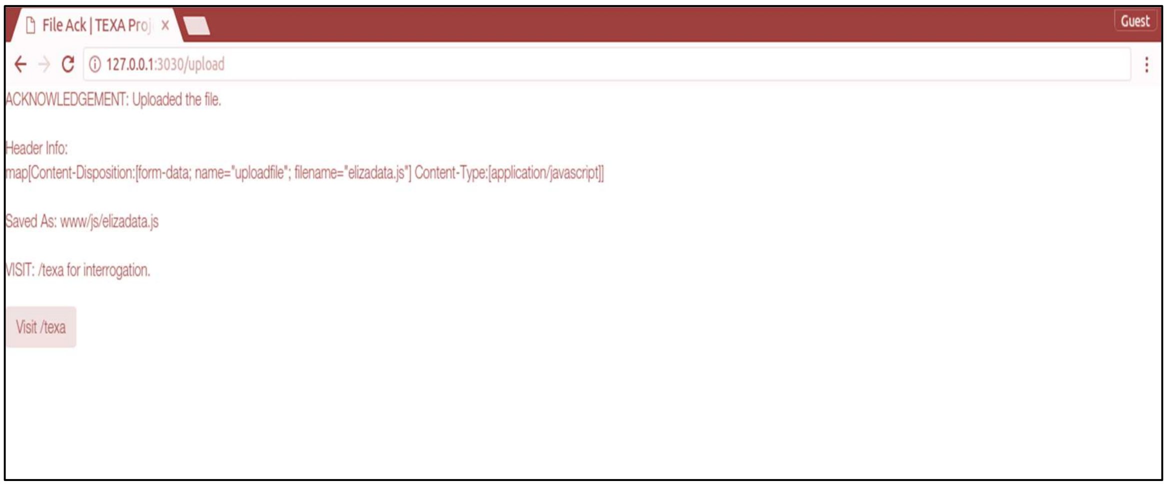
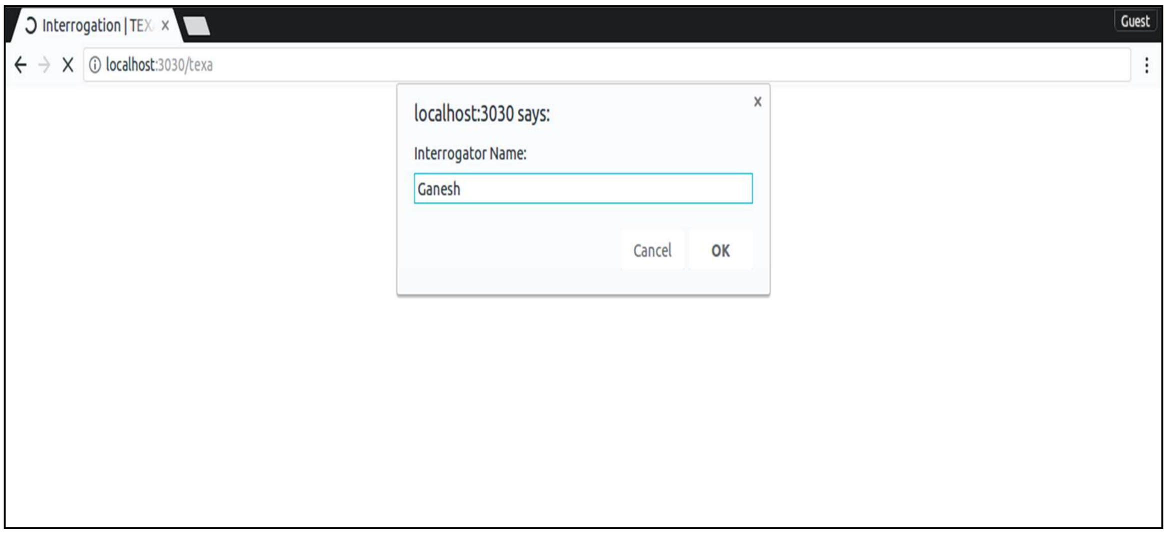


Fig 7.5: Dialog input to enter AI Name



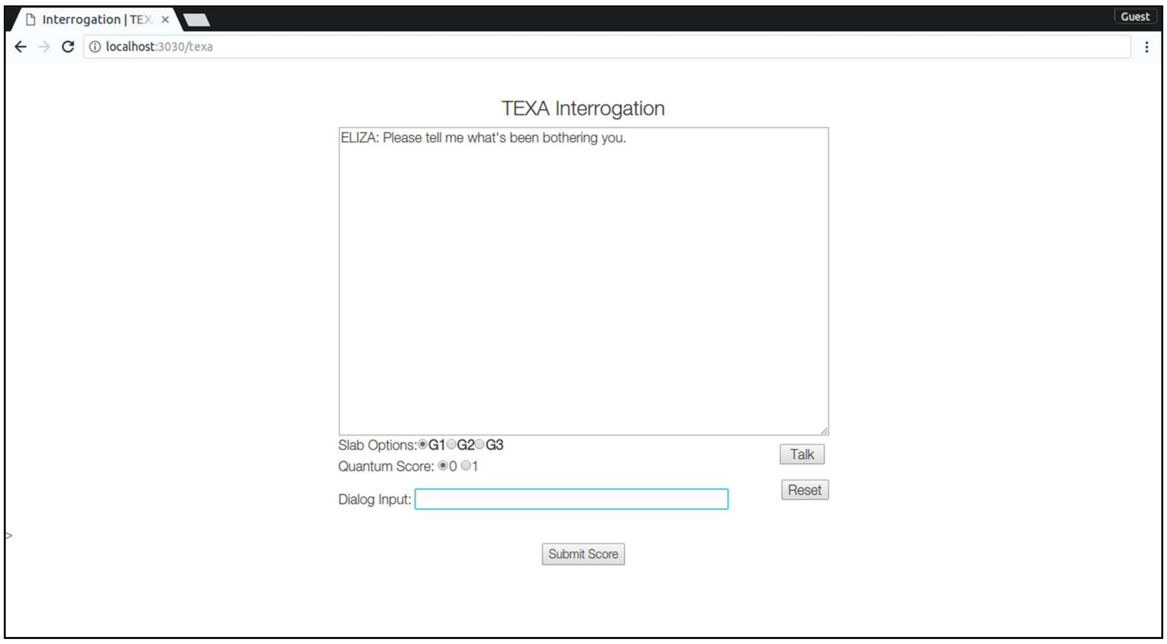
```
ACKNOWLEDGEMENT: Uploaded the file.

Header Info:
map[Content-Disposition:[form-data; name="uploadfile"; filename="elizadata.js"] Content-Type:[application/javascript]]

Saved As: www/js/elizadata.js

VISIT: /texa for interrogation.
```

Fig 7.6: Acknowledgement for AI file upload



localhost:3030 says:
Interrogator Name:
Ganesh

Fig 7.7: Dialog input to enter Interrogator Name

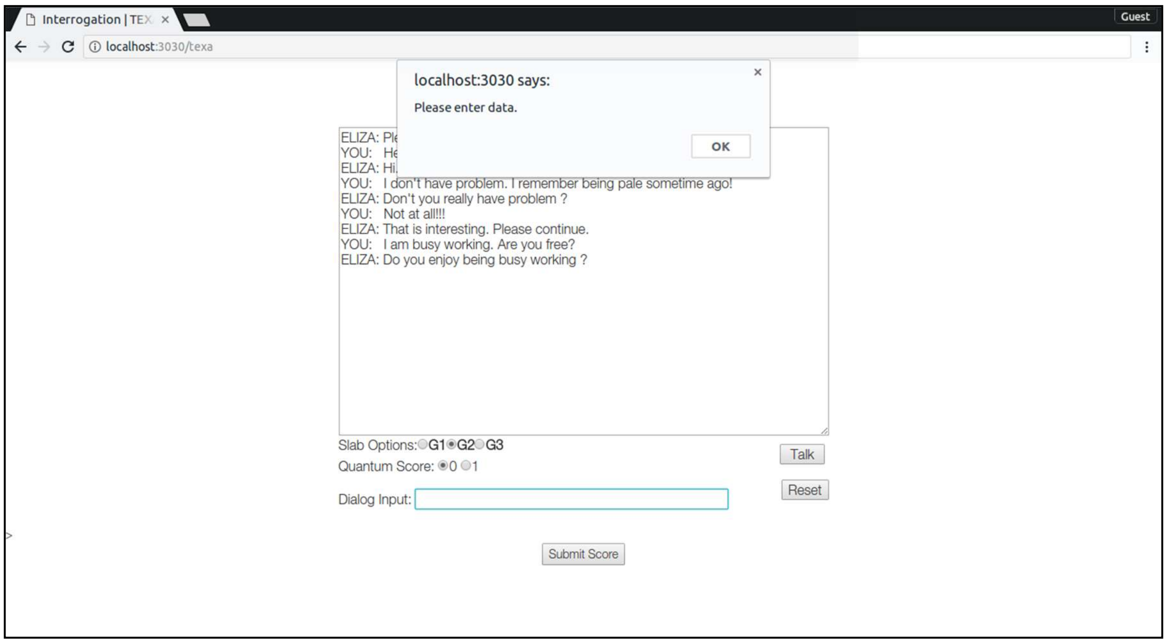


TEXA Interrogation

ELIZA: Please tell me what's been bothering you.

Slab Options: [✓ G1] G2 G3
Quantum Score: [✓ 0] 1
Dialog Input:

Fig 7.8: Initial snapshot of TEXA Interrogation



localhost:3030 says:

Please enter data.

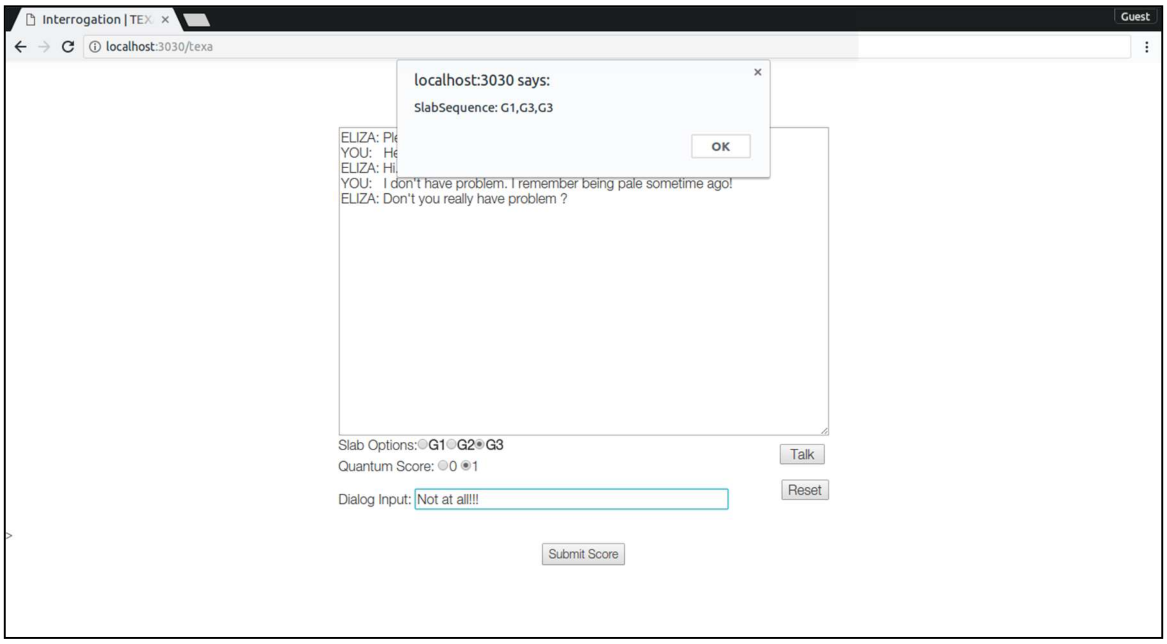
```
ELIZA: Pl
YOU:   He
ELIZA: Hi.
YOU:   I don't have problem. I remember being pale sometime ago!
ELIZA: Don't you really have problem ?
YOU:   Not at all!!!
ELIZA: That is interesting. Please continue.
YOU:   I am busy working. Are you free?
ELIZA: Do you enjoy being busy working ?
```

Slab Options: G1 [✓ G2] G3

Quantum Score: [✓ 0] 1

Dialog Input:

Fig 7.9: TEXA Interrogation recognizing empty dialog input



localhost:3030 says:

SlabSequence: G1,G3,G3

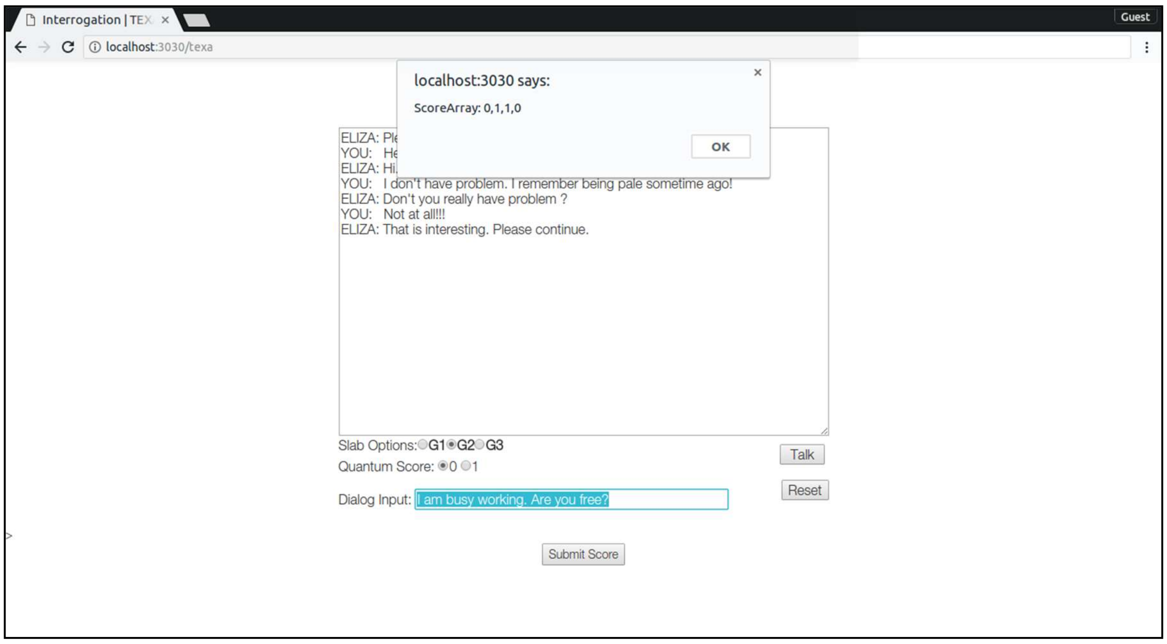
```
ELIZA: Pl
YOU:   He
ELIZA: Hi.
YOU:   I don't have problem. I remember being pale sometime ago!
ELIZA: Don't you really have problem ?
```

Slab Options: G1 G2 [✓ G3]

Quantum Score: 0 [✓ 1]

Dialog Input: Not at all!!!

Fig 7.10: TEXA Interrogation webpage acknowledging Slab Sequence Array



localhost:3030 says:

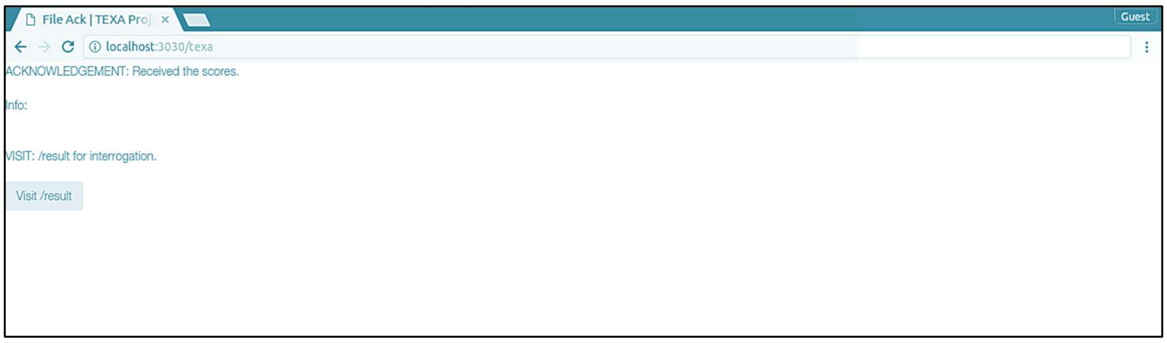
ScoreArray: 0,1,1,0

OK

ELIZA: Ple
YOU: He
ELIZA: Hi.
YOU: I don't have problem. I remember being pale sometime ago!
ELIZA: Don't you really have problem ?
YOU: Not at all!!!
ELIZA: That is interesting. Please continue.

Slab Options: G1 [✓ G2] G3
Quantum Score: [✓ 0] 1
Dialog Input: I am busy working. Are you free?

Fig 7.11: TEXA Interrogation webpage acknowledging Quantum Score Array

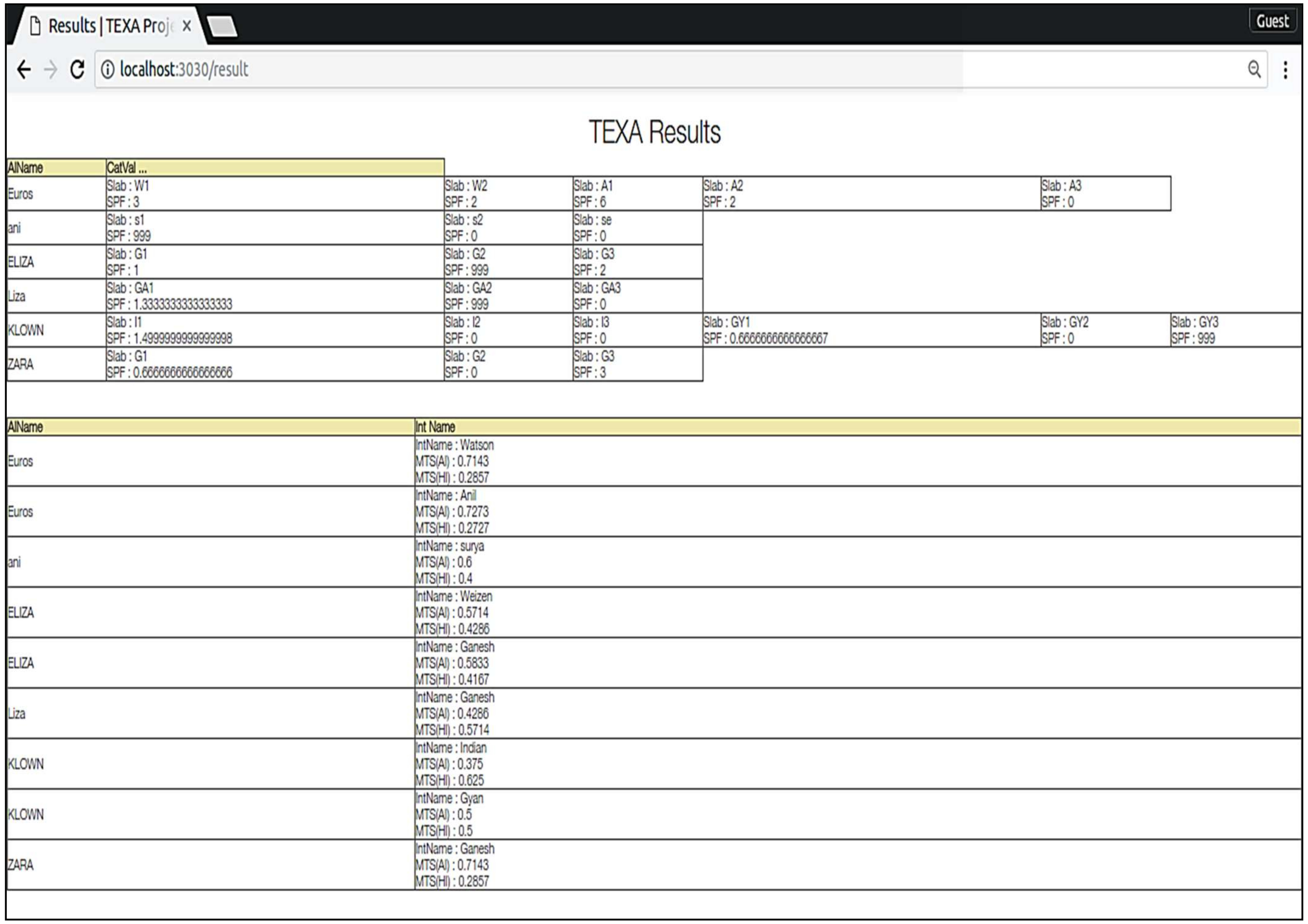


ACKNOWLEDGEMENT: Received the scores.

Info:

VISIT: /result for interrogation.

Fig 7.12: TEXA Interrogation acknowledging the SUBMIT action of Session data



Results | TEXA Proj

localhost:3030/result

TEXA Results

AIName	CatVal ...					
Euros	Slab : W1 SPF : 3	Slab : W2 SPF : 2	Slab : A1 SPF : 6	Slab : A2 SPF : 2	Slab : A3 SPF : 0	
ani	Slab : s1 SPF : 999	Slab : s2 SPF : 0	Slab : se SPF : 0			
ELIZA	Slab : G1 SPF : 1	Slab : G2 SPF : 999	Slab : G3 SPF : 2			
Liza	Slab : GA1 SPF : 1.3333333333333333	Slab : GA2 SPF : 999	Slab : GA3 SPF : 0			
KLOWN	Slab : I1 SPF : 1.4999999999999998	Slab : I2 SPF : 0	Slab : I3 SPF : 0	Slab : GY1 SPF : 0.6666666666666667	Slab : GY2 SPF : 0	Slab : GY3 SPF : 999
ZARA	Slab : G1 SPF : 0.6666666666666666	Slab : G2 SPF : 0	Slab : G3 SPF : 3			

AIName	Int Name
Euros	IntName : Watson MTS(AI) : 0.7143 MTS(HI) : 0.2857
Euros	IntName : Anil MTS(AI) : 0.7273 MTS(HI) : 0.2727
ani	IntName : surya MTS(AI) : 0.6 MTS(HI) : 0.4
ELIZA	IntName : Weizen MTS(AI) : 0.5714 MTS(HI) : 0.4286
ELIZA	IntName : Ganesh MTS(AI) : 0.5833 MTS(HI) : 0.4167
Liza	IntName : Ganesh MTS(AI) : 0.4286 MTS(HI) : 0.5714
KLOWN	IntName : Indian MTS(AI) : 0.375 MTS(HI) : 0.625
KLOWN	IntName : Gyan MTS(AI) : 0.5 MTS(HI) : 0.5
ZARA	IntName : Ganesh MTS(AI) : 0.7143 MTS(HI) : 0.2857

Fig 7.13: TEXA result webpage displaying result data of all the AIs' test instance

Chapter 8

CONCLUSION AND FUTURE SCOPE

8.1 Conclusion:

Several attempts made in the past to address the criticisms on the Turing Test hasn't been backed by mathematical models. Here, we have arrived at an exclusive and novel framework that doesn't only facilitate Turing test but also addresses key criticisms in order to broaden the applications.

8.2 Application:

Also, we observe that the data encompassed by the frameworks can be leveraged for multiple applications:

1. Analyzing and ranking commercial and academic AI
2. Using the standard score for betterment in the AI community
3. Engage in the same project and add new modes of interrogation and address important criticisms on early-stage AI, hence reducing the benchmark turnaround time.

8.3 Future Enhancements:

The exhibited TEXA prototype has a few limitations:

1. Stability offered only for Unit Test Instance. Can support multiple cases.
2. Lacks complete support for non-Eliza AIs.
3. APIs can be exposed to build use-cases such as ranking apps etc.

We welcome interests and initiatives to improvise the project with their best efforts. The entire project is open-sourced and available to general public at:

https://github.com/TexaProject/ (Pull requests are welcome!)

BIBLIOGRAPHY

[1] **A. M. Turing,** *Computing Machinery and Intelligence*. Mind 49: 433-460, 1950.

[2] **Katrina LaCurts**, *Criticisms of the Turing Test and Why You Should Ignore (Most of) Them*, MIT CSAIL, 6.893, 2013.

[3] **Joseph Weizenbaum**, *ELIZA - A Computer Program For the Study of Natural Language Communication Between Man And Machine*, Computational Linguistics Vol. 9 No. 1, 1966.

[4] **Wikipedia**, *Artificial Intelligence*, http://en.wikipedia.org/wiki/Artificial_Intelligence.

[5] **Wikipedia**, *Human Intelligence*, http://en.wikipedia.org/wiki/Human_Intelligence.

[6] **Wikipedia**, *Turing Test*, http://en.wikipedia.org/wiki/Turing_Test.

[7] **Wikipedia**, *Artificial Psychology*, http://en.wikipedia.org/wiki/Artificial_Psychology.