ABSTRACT

People in most of the time need some legal advice about various issues because they do not want to be subject of injustice, especially in their work life. Thus, they can be aware of how to deal with their administration and workmates. But these days is too expansive to pay for legal advice that might benefit them or not, and not all people have the ability. This project focuses on helping people to get free legal advice on labor area and can appoint and communicate with volunteered and artificial lawyers at any time. Also, users can read labor low's regulation to increase their knowledge. In order to solve this issue, we developed a web-based system that use a retrieval information chatbot which able to give appropriate labor's legal advice by answering user's questions using artificial intelligent (AI), natural language processing (NLP). Finally, the rustle of the system is guiding the citizens about their legal rights on the labor area by using different techniques.

Keywords: Free legal advice, Artificial intelligence, Information retrieval, Natural processing language, Chatbot.

TABLE OF CONTENT

ACKNOWLEDGMENTS V

ABSTRACT VI

LIST OF FIGURES X

LIST OF TABLE XII

CHAPTER 1 INTRODUCTION 1

1. INTRUDUCTION 2

1.1. PURPOSE OF THE PROJECT 2
1.2 PURPOSE OF THIS DOCUMENT 3
1.3 OVERVIEW OF THIS DOCUMENT 3
1.4 EXISTING SYSTEM 3
1.4.1 Free Legal Advice 3
1.4.2 ROSS 4
1.4.3 Ailira 4
1.4.4 Your Omaly consulter 5
1.4.5 DoNotPay 5
1.4.6 LISA 5
1.5 SUMMARY 5
1.6 BACKGROUND 7
1.6.1 Information retrieval chatbot 7
1.6.2 Natural language processing 7

CHAPTER 2 SYSTEM ANALYSIS 9

2. SYSTEM ANALYSIS 10

2.1 PROJECT SCOPE 10
2.2 SYSTEM REQUIREMENTS 10
2.2.1 Functional requirements 10
2.2.1.1 Manager should be able to 10
2.2.1.2 Lawyer should be able to 10
2.2.1.3 Client should be able to 10
2.2.1.4 Artificial lawyer 11
2.2.2 Data requirement 11
2.2.3 Non-functional requirements 11
2.2.3.1 Look and feel requirements 11
2.2.3.2 Usability requirements 11
2.2.3.3 Security requirements 12
2.2.3.4 Performance requirements 12
2.2.3.5 Availability 12

2.2.3.6 Cost requirements 12
2.2.3.7 Reliability 12
2.3 CONTEXT LEVEL DIAGRAM 13
2.4 DATA FLOW DIAGRAM 14
2.5 USE CASE DIAGRAM 19
2.6 USE CASE SCENARIO 20
2.7 PROPOSED SOLUTIONS 24
CHAPTER 3 DESIGN CONSIDERATIONS 29
3. DESIGN CONSIDERATIONS 30
3.1 DESIGN CONSTRAINTS 30
3.1.1 Software environment 30
3.1.1.1 Software environment 30
3.1.2 End-user characteristics 31
3.2 ARCHITECTURAL STRATEGIES 31
3.2.1 Algorithm to be used 31
3.2.2 Project management strategies 32
3.2.3 Development method 32
3.2.4 Architectural pattern 34
3.2.4.1 The Model-View-Controller pattern 34
CHAPTER 4 SYSTEM DESIGN 36
4. SYSTEM DESIGN 37
4.1 SYSTEM ARCHITECTURE AND PROGRAM FLOW 37
4.1.1 Major model 37
4.1.1.1 Web-based model 37
4.1.1.2 Database model 38
4.1.1.3 Chatbot model 38
4.1.2 Activity diagram 40
4.2 DETAILED SYSTEM DESIGN 43
4.2.1 Database details 43
4.2.2AAL system Interfaces 44
..... 50
FIGURE 27: LAWYER'S FORMS INTERFACE 50
FIGURE 28: LAWYERS LIST INTERFACE 50
FIGURE 29: HELP ME INTERFACE 51
FIGURE 30: SIGN UP BY FILL FORM INTERFACE 51
CHAPTER 5 IMPLEMENTATION 52
5. IMPLEMENTATION 53
5.1. AAL CHATBOT AGENT 53
5.1.1. Data pre-processing 53

5.1.2. AAL chatbot implementation 56
5.1.2.1. Prepare data for matching 56
5.1.2.2. Artificial lawyer pattern matching 60
5.2. AAL WEBSITE 65
5.2.1. LIVECHAT CONNECTION 65
5.2. LINKING CHATBOT TO AAL WEB-BASED SYSTEM 66
5.3. DEPLOYING AAL SYSTEM 67
5.3.1. AAL chatbot deployment 67
5.3.2. Web-based system deployment 69
CHAPTER 6 VALIDATION 70
6.VALIDATION 71
6.1 VALIDATION 71
6.1.1 ARTIFICIAL LAWYER EVALUATION 71
6.1.2 AAL HOSTING WEBPAGE VALIDATION 75
REFERENCE: 83

LIST OF FIGURES

Figure 1: context-level of AAL system 13
Figure 2: Manger data flow diagram 15
Figure 3: Lawyer data flow diagram..... 16
Figure 4: Client Data flow diagram 17
Figure 5: level 2 of communicate with Artificial lawyer process..... 18
Figure 6: level 2 of communicate with human lawyer process..... 18
Figure 7: Use case diagram for AAL system. 19
Figure 8: waterfall management strategy..... 32
Figure 9: MVC pattern..... 35
Figure 10: Entity relationship diagram 38
Figure 11: chatbot architecture design..... 39
Figure 12: Manager activity diagram..... 40
Figure 13: lawyer activity diagram..... 41
Figure 14: client activity diagram..... 42
Figure 15: Manager relation..... 43
Figure 16: Lawyer relation..... 43
Figure 17: Home interface 44
Figure 18: Artificial lawyer interface 45
Figure 19: Regulation labor law list interface..... 46
Figure 20: Regulation labor law interface 46
Figure 21: Lawyer list interface..... 47
Figure 22: Chat with lawyer interface 47
Figure 23: Manual help page interface 48
Figure 24: manger sign in interface 48
Figure 25: manger home page interface..... 49
Figure 26: manger update profile interface..... 49
Figure 27: lawyer‘s forms interface..... 50
Figure 28: lawyers list interface..... 50
Figure 29: help me interface 51
Figure 30: sign up by fill form interface..... 51
Figure 31: data cleaning code 54
Figure 32: Responses file..... 55
Figure 33: sample of response file. 56
Figure 34: Keywords list..... 57
Figure 35: Cleaned_keywords_Dic dictionary. 58
Figure 36: Keywords_patterns_dic dictionary..... 59
Figure 37: matching_counter_dic dictionary. 60
Figure 38: replacement dictionary. 60
Figure 39: clean_up_sentance function. 61
Figure 40: matching function..... 62
Figure 41: matching filtering function..... 63

Figure 42: Make_response_dic function. 64
Figure 43: response function. 65
Figure 44: LiveChat. 66
Figure 45: Flask code. 67
Figure 46: redeclare files path for Heruko server 67
Figure 47: requirments.txt file 68
Figure 48: Procfile file 68
Figure 49: deployment of AAL chatbot agent. 69
Figure 50: evaluation code. 72
Figure 51: first evaluation. 72
Figure 52: second evaluation. 73
Figure 53: Third evaluation. 74
Figure 54: fourth evaluation. 75
Figure 55: Lawyer volunteerism form interface. 76
Figure 56: Database for insert lawyer information. 76
Figure 57: Database for inserted lawyer information. 77
Figure 58: manager sign in validation. 78
Figure 59: Database before edit manager's profile. 78
Figure 60: Then deletion information messages will appear to the manager 79
Figure 61: Database after change manager's name 79
Figure 62: Database before deleting lawyer 79
Figure 63: Deletion confirmation messages 80
Figure 64: Then deletion information messages will appear to the manager 80
Figure 65: Database after deleting lawyer 80
Figure 66: database before accepting lawyer 80
Figure 67: view lawyer's request 81
Figure 68: lawyer's CV 81
Figure 69: database after accepting lawyer 81
Figure 70: accepting lawyer email 82

List of Table

Table 1: GLOSSARY OF TERM xiii
Tablet 2: comparison with related work 6
Table 3 use case scenario: Management manger account 20
Table 4 use case scenario: View lawyer form 21
Table 5 use case scenario: View accepted lawyer 20
Table 6 use case scenario: communicate with artificial lawyer. 21
Table 7 use case scenario: View law regulations 22

GLOSSARY OF TERM

Table 1: GLOSSARY OF TERM

Abbreviation	Its intended
AI	Artificial intelligence
ML	Machine learning
NLP	Natural language process
NLTK	Natural language toolkit
DB	Database

Chapter 1 INTRODUCTION

1. INTRUDUCTION

1.1. Purpose of the Project

People tend to use the Internet for different purposes including work, interaction with other people around the world, or just as an information source to learn more knowledge or seek offline consultations. Nowadays, especially after the initiation of the 2030 vision in KSA, it become common practice that people might volunteer to serve the community either by effort, knowledge, or money. Such movement in society requires a sophisticated platform that facilitates the interaction between the different parties. One of the potential cases, that we believe many citizens need legal advisory system where citizens can share their labor's cases with various lawyers who volunteering to provide free consultations and advices which upon law of Saudi constitution. We developed a web-based system that is responsible for ensuring respond to the request for advice on people's issues and guide them to excellent lawyers. As a result, we build a web-based system that facilitates anonymous interaction between citizens and lawyers to seek labor legal advices.

Furthermore, our intended system utilizes the concept of Artificial Inelegance (AI), natural language processing (NLP) and information retrieval chatbot to answer users' questions in the light of using pattern matching technique and find the most appropriate answers for them. Thus, the existing system able to answer the user who wrote their case in the system and inform them of the law/penalty in their case. The information retrieval chatbot is an artificial intelligent software which conducts a conversation via auditory or textual methods. Our chatbot built from scratch by using Python programming language and Elizabeth's pattern matching algorithm.

1.2 Purpose of this document

This document is intended to describe the details of the project conducted to develop the AAL system. It covers all the conceptual and technical aspects of the project and highlights the key analysis and findings of the work.

1.3 Overview of this document

This report divided into six chapters. Chapter 1 introduces the main idea of the project, describes the purpose of this project/document, reviews of existing systems, and describes similar systems to our existing system and background about machine learning and natural language processing. Chapter 2, system analysis chapter gives a detailed textual description of the system and depicts the types of users of the system, identifies functional and non-functional requirements, covers a number of UML diagrams, including Use-Case Diagram, Data-Flow Diagram and Context Diagram and provides proposed/alternative solutions. Chapter 3, design considerations, specifies the design constraints of hardware and software environment, and defines end user characteristics, Also, explains architectural strategies and focus on the algorithm to be used, methodology of development and architectural pattern. Chapter 4, system design, describes system architecture and program flow and detailed component description. Chapter 5, this chapter describes the implementation of AAL system using information retrieval chatbot. It describes the tools and way to implement the system. Chapter 6, This chapter describes the validation of AAL system using information retrieval chatbot it provides testing and training of the system.

1.4 Existing System

Many existing systems provide a solution to the same problem. In the following, we will discuss some of these systems.

1.4.1 Free Legal Advice

A free legal advice is web-based system which provide a free legal advice in English and it’s covered

all various legal aspects, when users able to have communicate with lawyers to have a free legal advice. First of all, users should choose the required legal section they want to consult about it then write their problem in a label text and send it as a request to the lawyers. The site also provides in each section frequently asked questions (FAQs) which may help users to find relevant answers to their cases [1].

1.4.2 ROSS

ROSS is a web-based system that provides a free legal using advice for law of the US constitution, using of AI to build artificially intelligent tools to enhance lawyers' abilities using NLP. Users able to upload their cases in written documents (.doc, .pdf) written in English then receive a summary that contain the advice [2].

1.4.3 Ailira

Ailira is web-based system developed by Cartland Law team which provides a free legal advice that following Cartland Law by using AI that uses NLP to provides the legal advice, including Business Structuring. That can be done either over a chatting session with a virtual agent that can provide preliminary guidance and generic information with fast and accurate performance that provide the advice in 30 second. Users also can communicate with a human lawyer in case users have specific requirements to discuss that the virtual agent is not able to handle [3].

1.4.4 Your Omaly consulter

Is web-based system that developed by Ministry of Labor and Development that supports Arabic language, users can have their legal advice by written their issues in label text and send it to lawyer team and the response will be sent to users in 3 days at least, also user can read the regulation in labor low to increase their own knowledge [4].

1.4.5 DoNotPay

Is the first robot lawyer was invented by British entrepreneur Joshua Browder. It's IOS app provides a free legal advice using chatbot, the users will answer a series of question that asked by chatbot then chatbot will provide to user a document that can be sent to courthouse [5].

1.4.6 LISA

Is web-based system that developed by British lawyers team up with DoNotPay[5] to create a robot lawyer that provide a legal help, so users can create their non-disclosure agreement (NDA) then LISA will help users to find a middle ground between them and the receiver and offers some guidance that helps users in their cases, basic legal questions will be answered, documents drafted and an agreement achieved on terms that are fair and reasonable for both sides[6].

1.5 Summary

Our survey highlighted that none of the solutions supported bot in the domain of Saudi Labor law. We have investigated the possibility to train the available systems to serve the domain of this project, but we found that all existing systems were closed

sourced without much details about how their ML engines might be trained and handled. Moreover, many of the solutions lack of Arabic language's support. Thus, this project built a web-based bot system to serve the described domain.

Tablet 2: comparison with related work

	Free legal advice	ROSS	Ailira	Your Omaly consulter	DoNotPay	LISA	AAL
Operates via website	✓	✓	✓	✓		✓	✓
Free legal advice	✓	✓	✓	✓	✓	✓	✓
Availability of volunteer lawyers							✓
Cover all legal aspects	✓	✓			✓	✓	
User can receive the legal advice instantly			✓		✓		✓
Ability to read regulation				✓			✓
Supports Arabic language				✓			✓

1.6 Background

1.6.1 Information retrieval chatbot

Artificial intelligence methods are employed throughout the standard information retrieval process. Information retrieval is a key technology for knowledge management. It deals with the search for information and the representation, it allows a user to retrieve one selected data item from knowledge base which already has been stored and organized. Information retrieval is concerned with search processes in which a user needs to identify a subset of information which is relevant for his information need within a large amount of knowledge. [7][8]

The process of information retrieval starts when a user creates any query, it doesn't match to the right data object instead it matches with the several collections of data objects for further evaluation. The ranking of relevant documents is done to find out the most related document to the given query, this is the difference between the database searching and information retrieval. [9]

After the query is sent to the core of the system. This part has the access to the content management module which is directly linked with the back-end i.e. the large collections of data object, once results are generated by the core system then it is returned to the user by some graphical user interfaces. The process repeats until the user satisfied for what he is actually looking for. [9]

1.6.2 Natural language processing

We used natural language processing (NLP) in our project which gives the ability to machine for handling natural languages as effectively as humans and beings understanding the language [10].

For processing NLP we used the Natural Language Toolkit (NLTK) is a suite for program modules, data sets, tutorials and exercises, covering symbolic and statistical natural language processing. NLTK is a broad-coverage provides a simple, extensible, uniform framework for assignments, demonstrations, and projects. It is thoroughly documented, easy to learn, and simple to use. NLTK is now widely used in

research and teaching [11]. We chose this library upon others because it provides a flexible framework for our project. Typical projects might involve implementing a new algorithm, developing a new component, or implementing a new task. Further NLTK is written in Python which we are familiar with. NLTK is implemented as a large collection of minimally interdependent modules, organized into a shallow hierarchy. A set of core modules defines basic data types that are used throughout the toolkit. One of these modules in nltk.tokenizer module that devoted to the task of tokenizing, or dividing a text into its constituent parts [12] as implemented in our project. Then import nltk.stem.isri Arabic stemmer to strip affixes from the token and return the stem. Furthermore nltk.corpus import stopwords to filtering the text from stopwords.

Chapter 2 SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

This chapter presents the analysis of the AAL system. It starts by describing the project scope, then discusses the types of users who are going to deal with the system, it lists functional and nonfunctional requirements of the system and finally context level, data flow and use case diagrams.

2.1 Project scope

The scope of this project is to design and develop a web-based system that can assist the citizens of Saudi Arabia to explore and understand the labor law as per the Saudi regulations. Furthermore, the system provided the capability to handle user's enquiries for the cases that are not listed in the regulations or need further investigation.

2.2 System requirements

In this section we will clarify the functional requirement, data requirement and finally non-functional requirement.

2.2.1 Functional requirements

This section will clarify the functional requirement for each user manager, lawyer, client, Artificial lawyer.

2.2.1.1 Manager should be able to

- Update her/his profile.
- View lawyer forms.
- Accept/decline lawyer.
- Delete lawyers.

2.2.1.2 Lawyer should be able to

- Submit a volunteering request

2.2.1.3 Client should be able to

- View lawyers list.
- Communicate with human lawyer.

- Communicate with Artificial lawyer.
- View laws regulation page.

2.2.1.4 Artificial lawyer

- The output of Artificial lawyer should be text.
- The Artificial lawyer should response for any client input.
- if it doesn't understand the client's case it should be asked for explanation.

2.2.2 Data requirement

The data resource is the labor law that provided by Ministry of Labor and Development [13] so there's no need for data collecting.

Pre-processing is needed for the data, it must be cleaned and analyzed to be used in the information retriever chatbot either for training data set or client's inputs, cleaning have been done by using NLP which processed by NLTK library, so data tokenized, removed punctuation, removed the non-alpha, removed stop word and stemming.

2.2.3 Non-functional requirements

Non-functional requirements are specifying the system's quality attributes and its criteria regarding look and feel, usability, security, performance requirements. These requirements and others have carefully described in the following points:

2.2.3.1 Look and feel requirements

- The messaging is a familiar way to the client.
- Metaphors are clear and simple

2.2.3.2 Usability requirements

- The system is easy to use for users.

- We provided the regulations laws for clients.
- Client doesn't need to create account to access to system's services.
- The client will remember how to use, once learned.

2.2.3.3 Security requirements

- The chat with artificial lawyer will be deleted after terminating.
- The system implements SHA-256 security algorithm in entering the password to login to the system.
- The massages between client and lawyer through LiveChat will be encrypted.

2.2.3.4 Performance requirements

- The system is fast in responding to clients' cases.
- The performance of the system is good in wide range of users that access the system concurrently.

2.2.3.5 Availability

- The system is available anytime to take an advice and able to receive many requests without any fails and retrieve the results in a short amount of time.

2.2.3.6 Cost requirements

- The system is open sources and can be used freely without any subscription charges.

2.2.3.7 Reliability

- The system avoid suddenly crashes after several usages.

- The system provided the correct answers related to each clients' issue.

2.3 Context level diagram

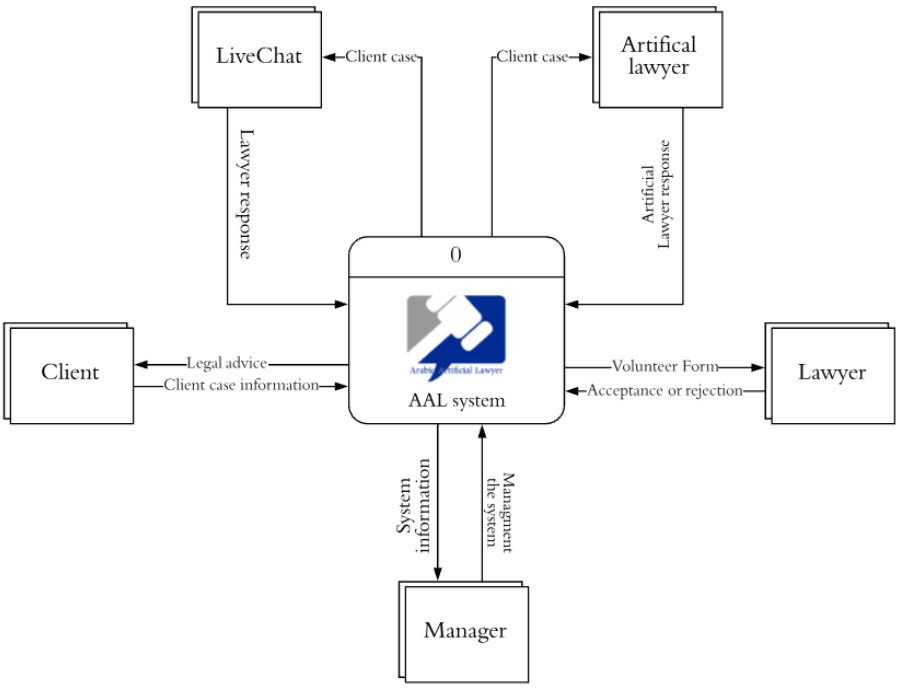
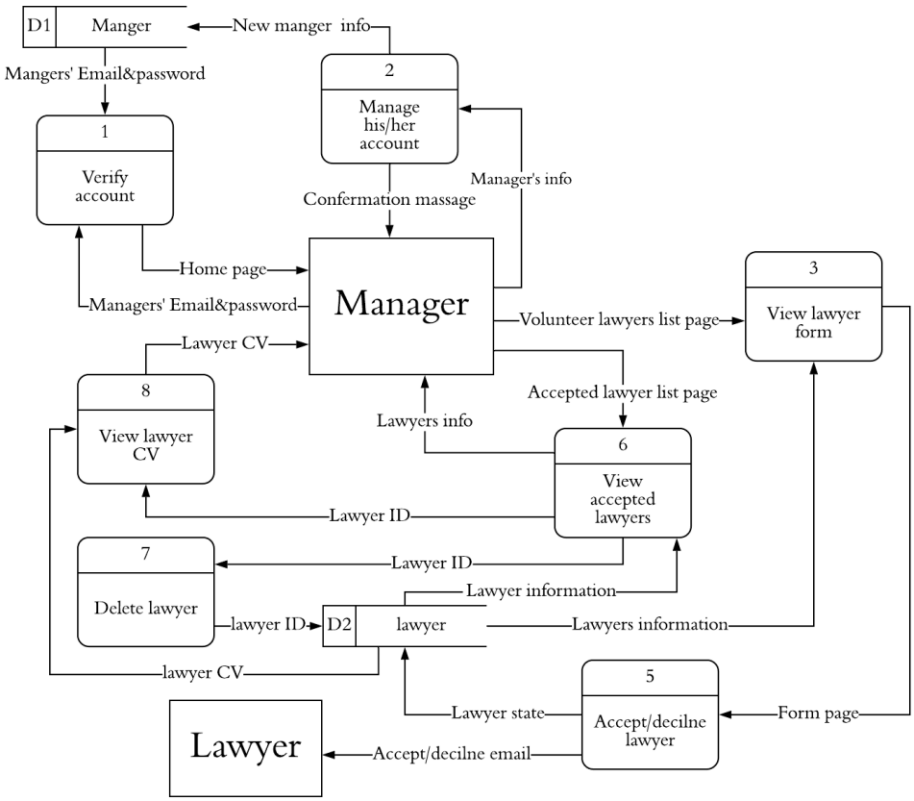


Figure 1: context-level of AAL system

2.4 Data flow diagram

- **Manager**



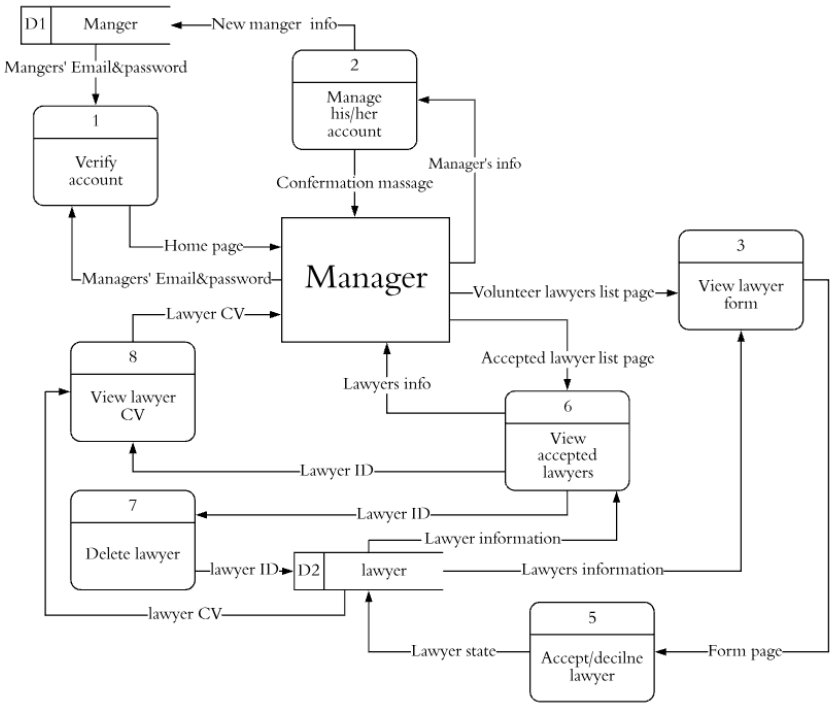


Figure 2: Manger data flow diagram

- **Lawyer**

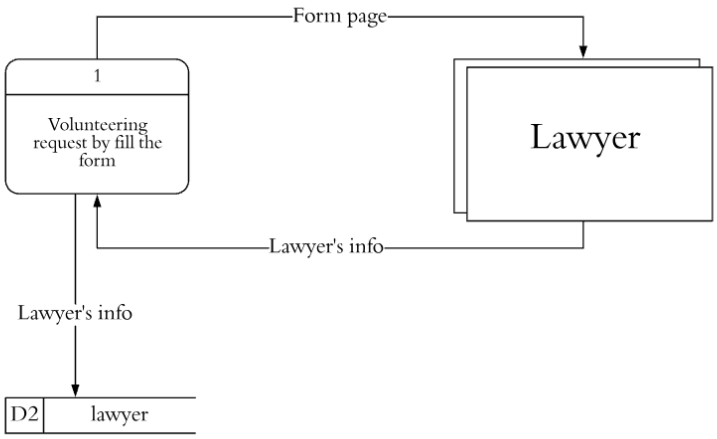


Figure 3: Lawyer data flow diagram

- **Client**

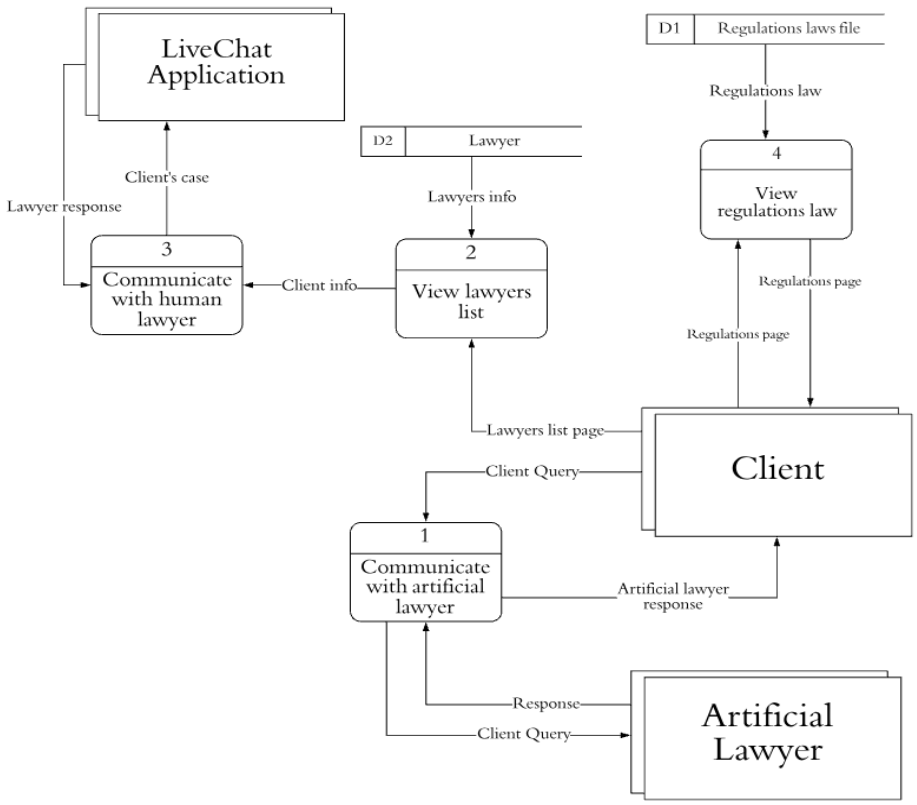


Figure 4: Client Data flow diagram

- **Communicate with artificial lawyer process**

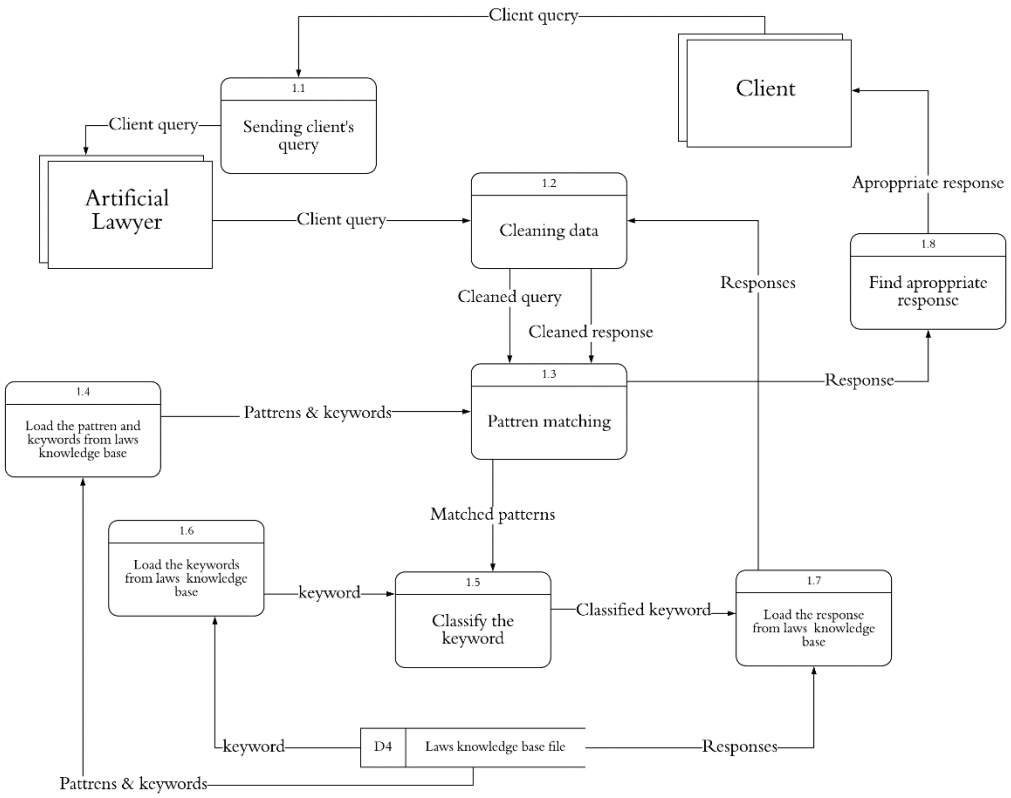


Figure 5: level 2 of communicate with Artificial lawyer process

- **Communicate with human lawyer process**

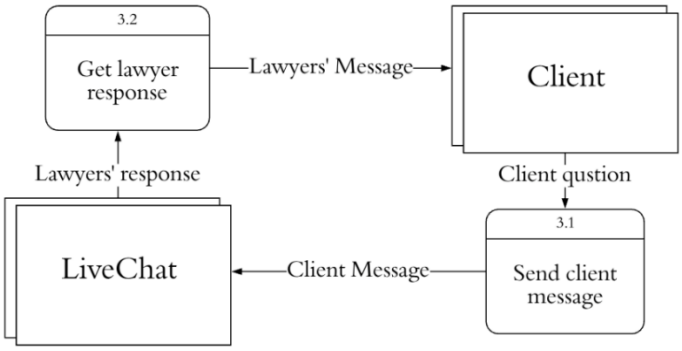


Figure 6: level 2 of communicate with human lawyer process

2.5 Use case diagram

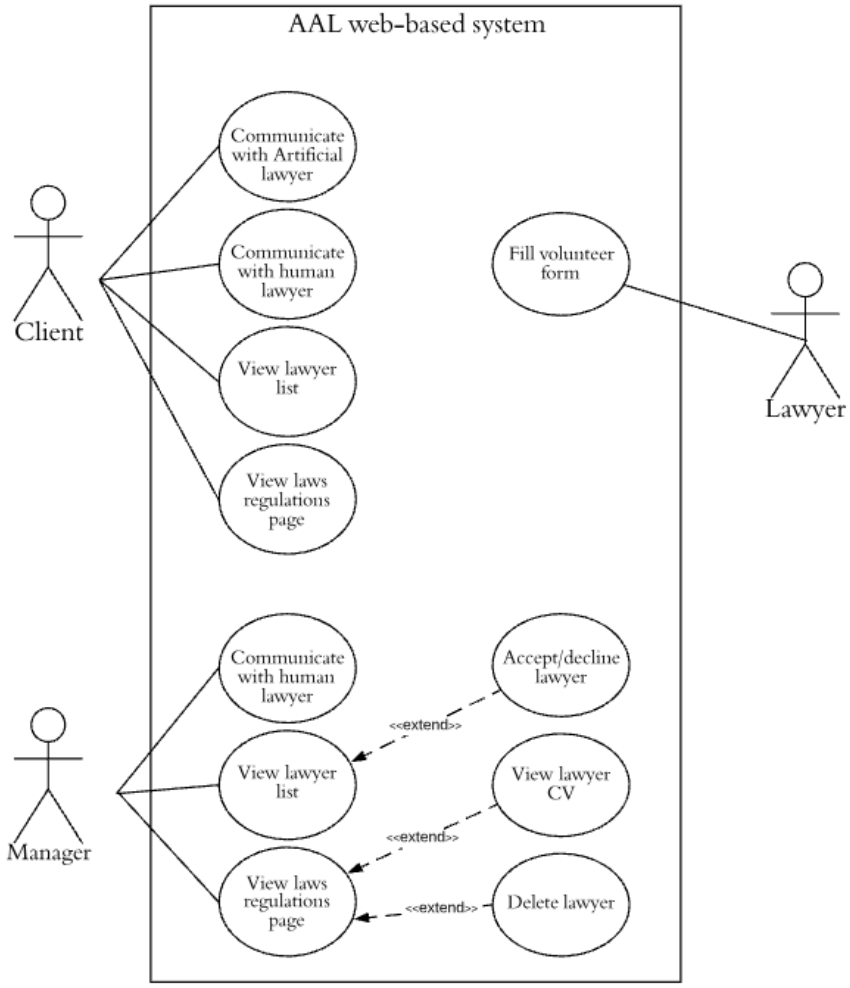


Figure 7: Use case diagram for AAL system.

2.6 Use case scenario

- **Management manger account**

Table 3 use case scenario: Management manger account.

Use case ID	A01
Use Case Title	Management manager account.
Primary Actor	Manager.
Brief Description	Allow manger to sign in, edit his/her profile.
Pre-Condition	Manager is already having an account.
Flow of Events	a. System will display sign in interface to enter his/her info (email & password) [A1]. b. System will validate info from manager database. [E1] c. Successfully sign in. d. The use case ends.
Alternative Flow	[A1] manger already signed in and want to edit his/her profile: a. The system display interface to allow manger to update relevant info. b. The system will check new entered info. [E2] c. The system saves new info and update it into manager database. d. The use case ends.
Exceptional Flow	[E1] Invalid email/password. a. The system displays an invalid email/password. b. The manager acknowledges the error. c. The system requests the email/password.

	d. The use case continues. [E2] Used invalid info. a. The system displays info are used. b. The manager acknowledges the error. c. The system requests new info. d. The use case continues.
Post-Condition	The system will view home page.

- **View lawyer form**

Table 4 use case scenario: View lawyer form.

Use case ID	A02
Use Case Title	View lawyer form.
Primary Actor	Manager.
Brief Description	Allow manager to view lawyer form to accept or decline it.
Pre-Condition	Manager is already sign in.
Flow of Events	a. The system shows lawyers' forms for manager. b. The system allows manager to accept or decline lawyers' forms request. [A1] [A2] c. The use case ends.
Alternative Flow	[A1] Manager accept lawyer's form request: a. Manager accept lawyer's form then submit the choice. [E1] b. The system will upload lawyers' info from form that has been accepted by manager to lawyer database.

	c. The manager will send an email to lawyer to confirm that he/she has been accepted. d. The use case ends. [A2] Manager decline lawyer's form: a. Manager decline lawyer's form then submit the choice. [E1] b. The use case ends.
Post-Condition	View lawyers' info from database if they accepted.

- **View accepted lawyer**

Table 5 use case scenario: View accepted lawyer.

Use case ID	A03
Use Case Title	View accepted lawyer.
Primary Actor	Manager.
Brief Description	Allow manager to view accepted lawyer information.
Pre-Condition	Manager is already sign in.
Flow of Events	a. The system shows lawyers' information for manager. b. The system allows manager to delete lawyers' or view their CVs. [A1] [A2] c. The use case ends.
Alternative Flow	[A1] Manager delete lawyer from database: a. Manager delete lawyer's information form database. [E1] b. The use case ends.

	[A2] Manager view lawyer's CV: a. Manager have access to view lawyer's CV. b. The use case ends.
Exceptional Flow	[E1] Wrong choice: a. The system will display a new page for manager to confirm delete choice. b. The use case continues.
Post-Condition	View lawyers' info from database if they accepted.

- **Communicate with Artificial lawyer**

Table 6 use case scenario: communicate with artificial lawyer.

Use case ID	A04
Use Case Title	Communicate with Artificial lawyer
Primary Actor	Client
Brief Description	Allow client to communicate with Artificial lawyer by asking a question and get responds.
Pre-Condition	Client must be online and using Arabic language.
Flow of Events	a. System will display Artificial lawyer interface. b. Client will type his/her case and send it. c. Artificial lawyer will respond to the client. [A] d. Use case end.
Alternative Flow	[A1] If Artificial lawyer doesn't find the answer: a. Artificial lawyer will ask for more explanations.
Exceptional Flow	Not applicable.
Post-Condition	Client will get the legal advice.

- **View labor law regulation**

Table 7 use case scenario: View labor law regulation.

Use case ID	A05
Use Case Title	View laws regulation page
Primary Actor	Client
Brief Description	Allow client to view labor law regulation.
Pre-Condition	Client must be online
Flow of Events	a. System will view a list of labor law regulation. b. Client can choose wanted labor law regulation. c. The system will retrieve the labor law form a text file. [E1] d. System will display labor law regulation. e. Use case end.
Alternative Flow	Not applicable.
Exceptional Flow	[E1] if system couldn't load law for file: a. Reload law from file. b. Use case continues.
Post-Condition	The system will display required regulation.

2.7 Proposed Solutions

The proposed solution of this project is to develop an agent (i.e. artificial lawyer) that interact with clients in order to provide them with the necessary information they might need regarding their cases. For more advanced cases that our agent may not handle the system will propose a number of human lawyers those might be available to provide suggestions.

Chapter 3 DESIGN CONSIDERATIONS

3. DESIGN CONSIDERATIONS

This chapter presents the design constraints of AAL system, starting by defining the requirements of software, then describing end user characteristics, discussing the algorithm that used, methodology to develop AAL system and the architectural pattern.

3.1 Design constraints

3.1.1 Software environment

To implementation and generation of AAL system there's some environments need to be used.

3.1.1.1 Software environment

- **Mobirise:** is a free offline app for Windows and Mac to easily create small/medium websites, landing pages, online resumes and portfolios. It's provided website blocks, templates and themes that's helps to build our site [14].
- **MYSQL Database:** is an open source relational database management system, used to store the managers' and lawyers' information [15].
- **HTML & JavaScript & PHP:** we used HTML, JavaScript and PHP programming languages to develop AAL system.
- **Python 3.6:** we used python programming language to develop information retrieval chatbot model.
- **Anaconda3:** is open source and free distribution that could be used for python language. It's could be used for developing, testing, and training on a single machine [16]. We used Anaconda3 to build our information retrieval chatbot agent to process and handle clients' queries.

- **LiveChat:** is an online customer service software with live support, help desk software and web analytics capabilities used to communicate with human lawyers [17].

3.1.2 End-user characteristics

The types of actors those might interact with the AAL system are: manager, lawyer and client. The manger actor is responsible for administrating and monitoring the system. The lawyer is an actor who might provide legal advice for labor law cases and responds to client's inquiries. The client actor is anyone who needs legal advice for labor law cases and might use the AAL system to seek consultations.

3.2 Architectural strategies

3.2.1 Algorithm to be used

We reused Elizabeth's Pattern matching algorithm [18] [19] which have been used with Elizabeth chatbot that has the same mechanism of pattern-matching that used in our AAL chatbot. But we've been add some modifications to make it more suitable for our chatbot.

The Elizabeth algorithm stages with our modifications in the following steps:

- Stage one: clean client input using NLTK library.
- Stage two: keyword pattern matching, all keywords are applied in turns until a match is found, then go stage three. The response is giving according to the best match, so if no match at all found or there is three or more of keyword matching, then go to stage four [18] [19].

- Stage three: if there is a match found, then one of the responses that related to the keyword will be selected based on best matching.
- Stage four: if no match occurs or there is three or more of keyword matching, one of the chatbot questions will be selected randomly [18] [19].

3.2.2 Project management strategies

We followed the waterfall [20] process model to define the different stages for developing AAL System. We decided to select waterfall model as the requirements for building AAL is clear and stable, but one of waterfall risk the development process will move to the next phase and there is no turning back to previous phase and there is no overlapping in the phases. The different stages of our project are described in Figure 8.

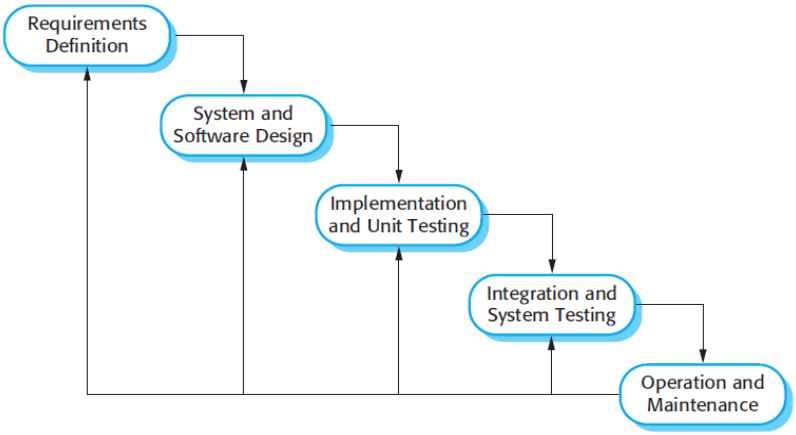


Figure 8: waterfall management strategy

3.2.3 Development method

Waterfall model is a sequential model consider as a plane-driven model that divides software development into different phases. Each phase is designed for performing specific activity during SDLC phase. These phases mainly consist of five phases:

- **Analysis**: In this phase, we have to specify the requirements of the proposed system such as:
  - The main features and functionalities.
  - Tools and dataset that's help us to build the system.
- **Design**: In this phase, we:
  - Developed a solution that analyzed from the analysis phase.
  - Draw all the diagrams that meet the requirements.
  - We built an information retriever chatbot engine.

- **Implementation**

  In implementation phase, we followed a sequence of steps such as:
  - We started the implementation with none cleaning data for labor law that provided by Ministry of Labor and Development [13].
  - We cleaned and analyzed data that provided by Ministry of Labor and Development [13].
  - We Implemented Elizabeth's Pattern matching algorithm [18] [19] for the chatbot behavior
  - We built interfaces for communication between artificial lawyer with client.

- **Testing:**

  After done with all implementation steps:
  - We evaluated the information retrieval chatbot by different experiment.
  - The system tested to obtain the validation and verification of the operations and functionalities.
  - The programming pugs also have been removed.

3.2.4 Architectural pattern

Patterns are mean of use representing, sharing, reusing knowledge and stylized descriptions of good design practice, which has been tried and tested in different environments.

3.2.4.1 The Model-View-Controller pattern

MVC Pattern stands for Model-View-Controller Pattern. In general, an MVC description shows a possible run-time architecture when this pattern is used for interaction management in a web-based system as shown in Figure 9.

This model has a tree logical component which interact with each other:

a. The Model component manages the system data and associated operations on that data. Therefore, model components represent local database, knowledge base and external service.

b. The Controller component manages user interaction (e.g., button, typing text in text label, mouse click etc.) and passes these interactions to the View and the Model.

c. The View component defines and manages how the data is presented to the users and the admins of the system [21].

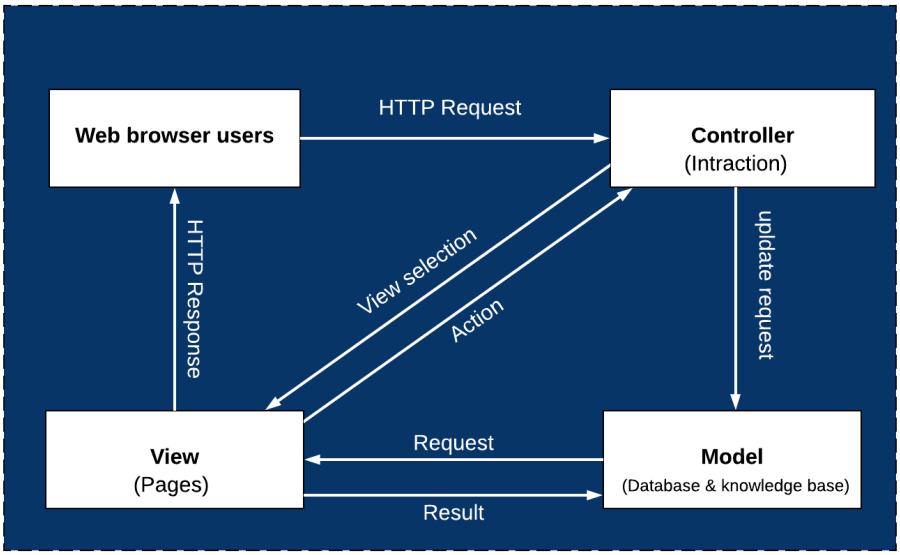


Figure 9: MVC pattern

Chapter 4 SYSTEM DESIGN

4. SYSTEM DESIGN

After specified and defined the requirements, we got a big picture of AAL system's processes. In this chapter, we specified the design of the system in the major model, activity diagram, database structure, and the interfaces of the AAL system.

4.1 System Architecture and Program Flow

4.1.1 Major model

This section describes the major model of AAL system which contains three models, naming, web-based model, database model and information retrieval chatbot model.

4.1.1.1 Web-based model

- **Front-End**

The front-end of the website is the part that users interact with. Everything that you see when you are navigating around our site, from fonts and colors to dropdown menus and sliders, is a combo of HTML, CSS and JavaScript being controlled by mobiries templets [14].

- **Back-End**

The back-end side of our website consists of a server and database. In order to make the server and database communicate with each other, back-end devs use server-side languages like PHP and tools like MySql to find, save, or change data and serve it back to the users in front-end code and allows us to manipulate the source code.

4.1.1.2 Database model

AAL system used the MySQL Workbench database which is a database that contains the flowing relations. (Lawyer, Manager). As shown in Figure 10 below.

- **Entity relationship diagram**

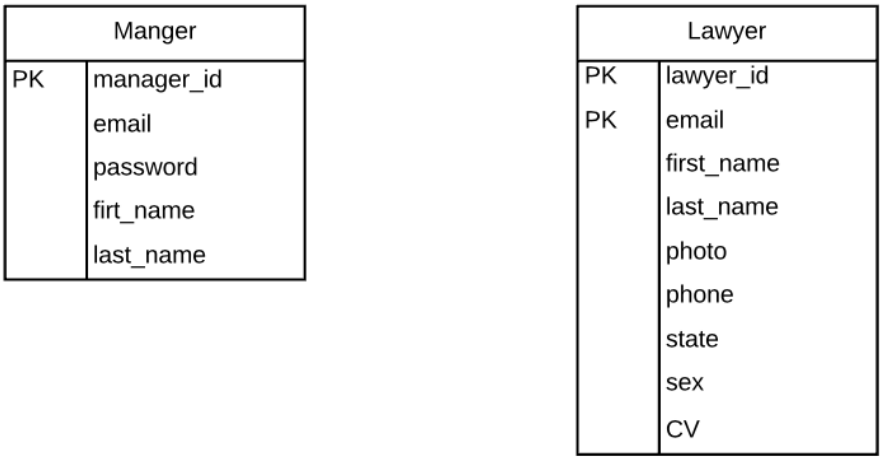


Figure 10: Entity relationship diagram

4.1.1.3 Chatbot model

We used pattern matching technique to generating the appropriate response to the client; we clarify the steps of chatbot model through information retrieval chatbot architecture design which presented in the Figure 11 below. First, the user query has been cleaned then the artificial lawyer will match it with patterns of the keywords and pattrens from knowledge base, then the keyword classifier identify the keyword of client query that will be done by selecting one of predefined keyword. By using the keyword, the system will generate a set of responses from

knowledge based of AAL system. Then, the response selector will choose an appropriate response based on the best matching process[22].

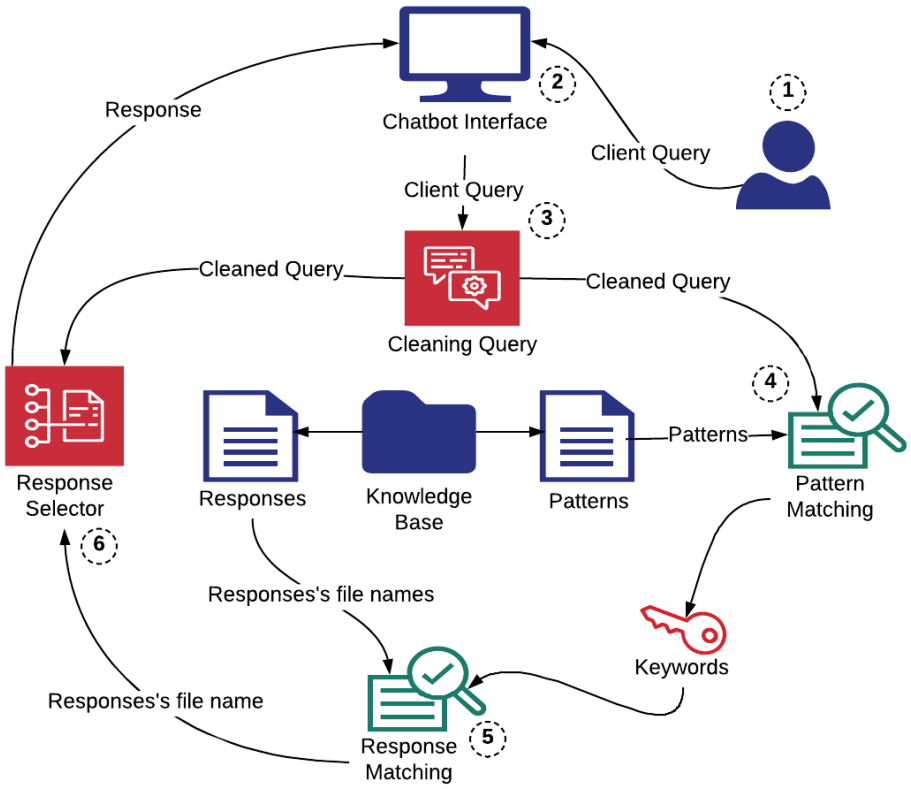


Figure 11: chatbot architecture design

4.1.2 Activity diagram

- **Manager**

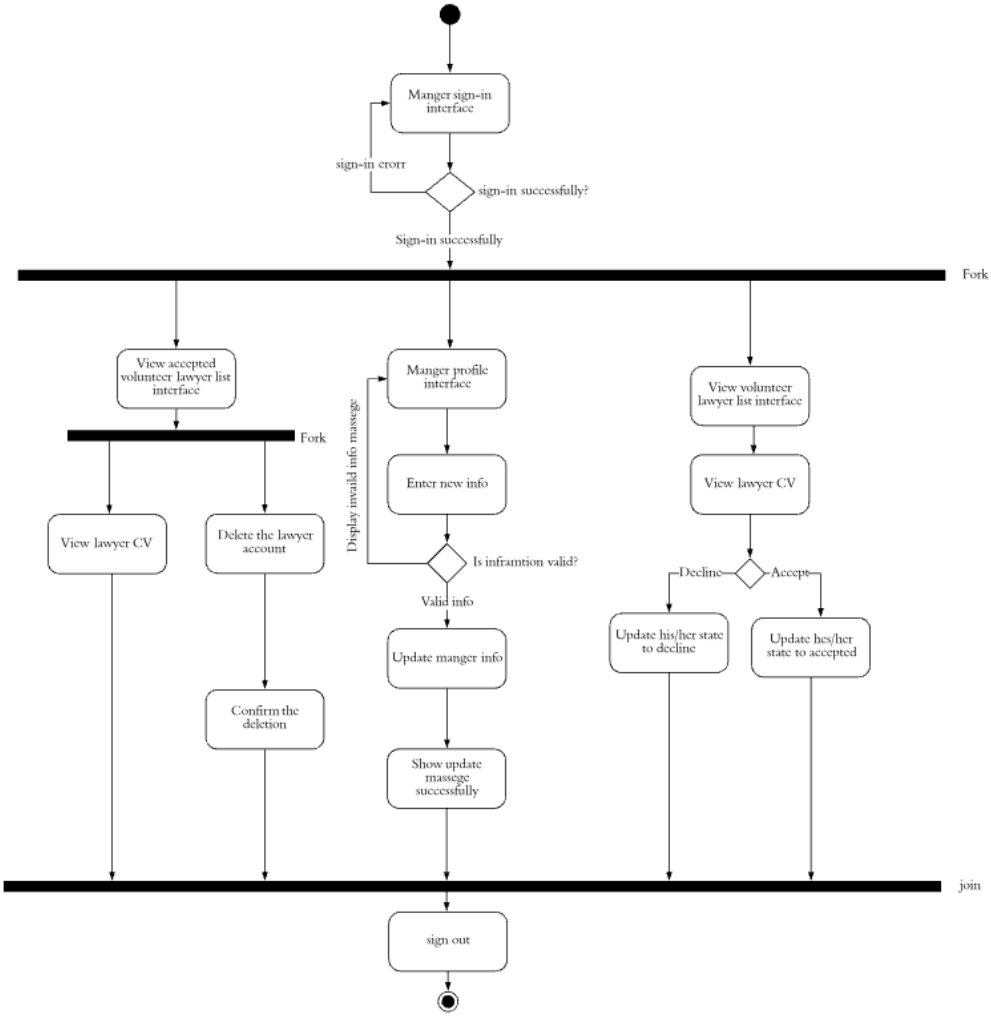


Figure 12: Manager activity diagram

- **Lawyer**

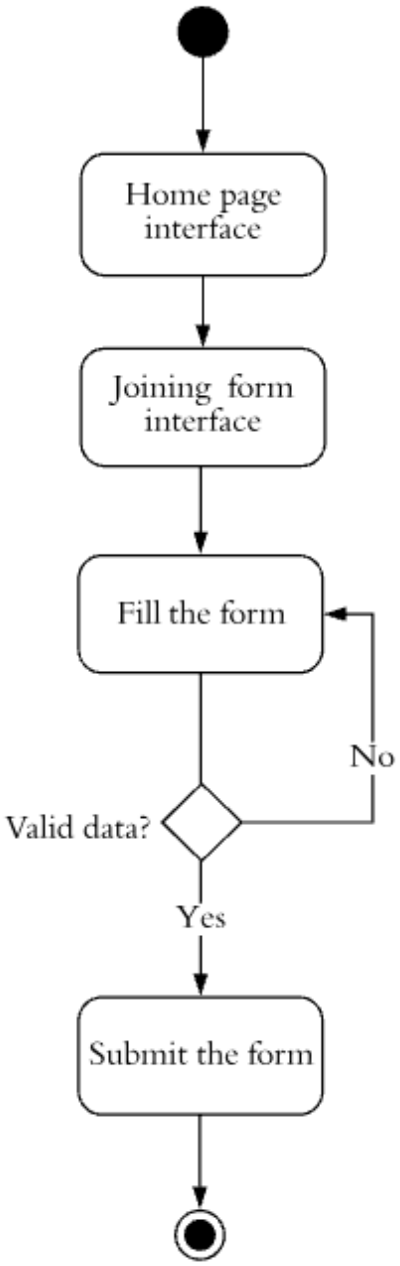


Figure 13: lawyer activity diagram

- **Client**

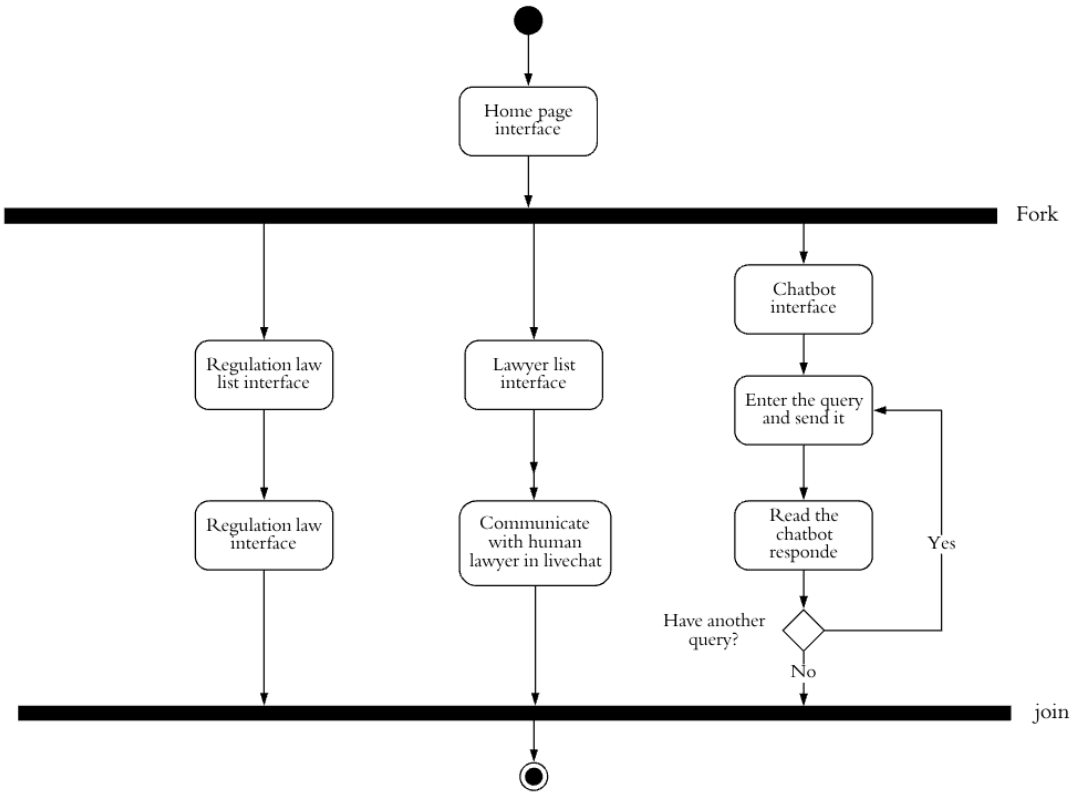


Figure 14: client activity diagram

4.2 Detailed System Design

4.2.1 Database details

- **Manager relation**

Figure 15 below shows Manager relation in DB contains five attributs manger_id as primary key, email, password, first_name and last_name.

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G
manager_id	VARCHAR(25)	☑	☑	☐	☐	☐	☐	☐	☐
manager_email	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐
password	VARCHAR(64)	☐	☑	☐	☐	☐	☐	☐	☐
first_name	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐
last_name	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐
<click to edit>		☐	☐	☐	☐	☐	☐	☐	☐

Figure 15: Manager relation

- **Lawyer relation**

Figure 16 below shows lawyer table in DB contains nine attributes lawyer_email as primary key, first_name,last_name, lawyer_id, state, sex, CV, phone, photo).

Name: lawyer Schem

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G
lawyer_email	VARCHAR(45)	☑	☑	☐	☐	☐	☐	☐	☐
first_name	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐
last_name	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐
photo	TEXT	☐	☐	☐	☐	☐	☐	☐	☐
phone	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐
state	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐
sex	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐
cv	TEXT	☐	☑	☐	☐	☐	☐	☐	☐
lawyer_id	VARCHAR(45)	☑	☑	☑	☐	☐	☐	☐	☐
<click to edit>		☐	☐	☐	☐	☐	☐	☐	☐

Figure 16: Lawyer relation

4.2.2 AAL system Interfaces

In this section, we described the system's interfaces with different views: manger view, client view.

- **Client view**
  - **Home page:** this page will be showed for all users when different main service of AAL system is listed, users can use chatbot and take their legal advice, as shown in Figure 17 below.

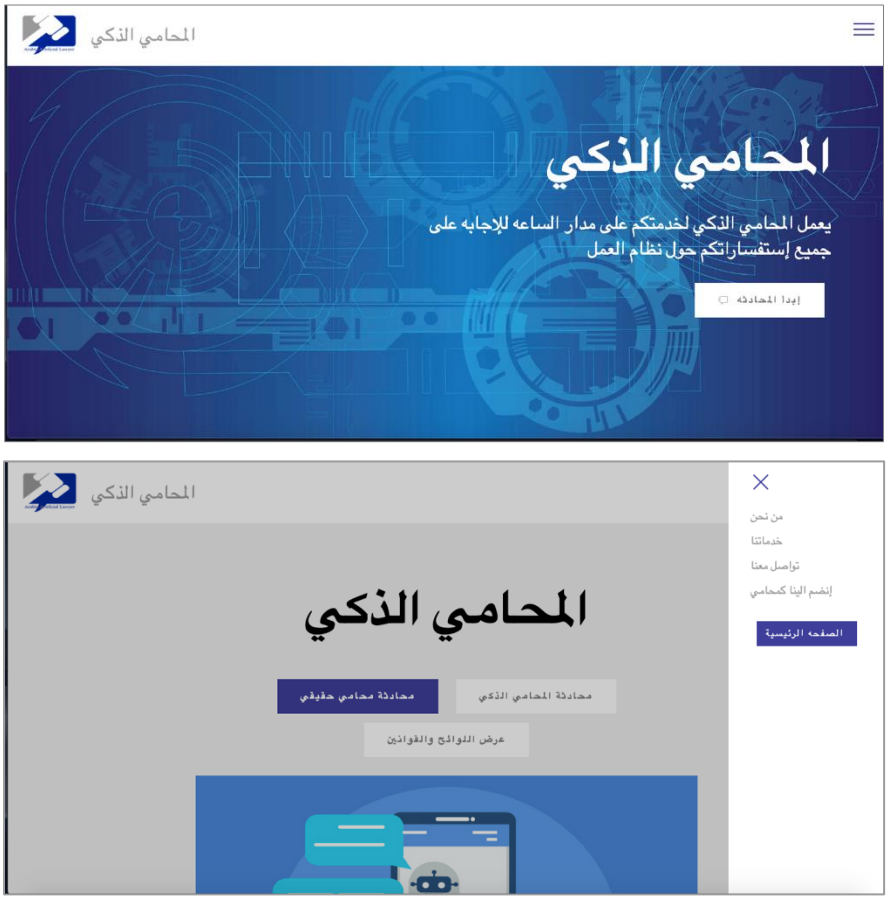


Figure 17: Home interface

- **Artificial lawyer interface:** in this page designed to be simple and familiar interface, so clients can used it easily, as show in Figure 18 below.

Figure 18: Artificial lawyer interface

- **Regulation labor law list interface**: in this page view the regulation of labor law as a list, thus, clients choose one of it to read more about interested labor law, as shown in Figure 19 below.

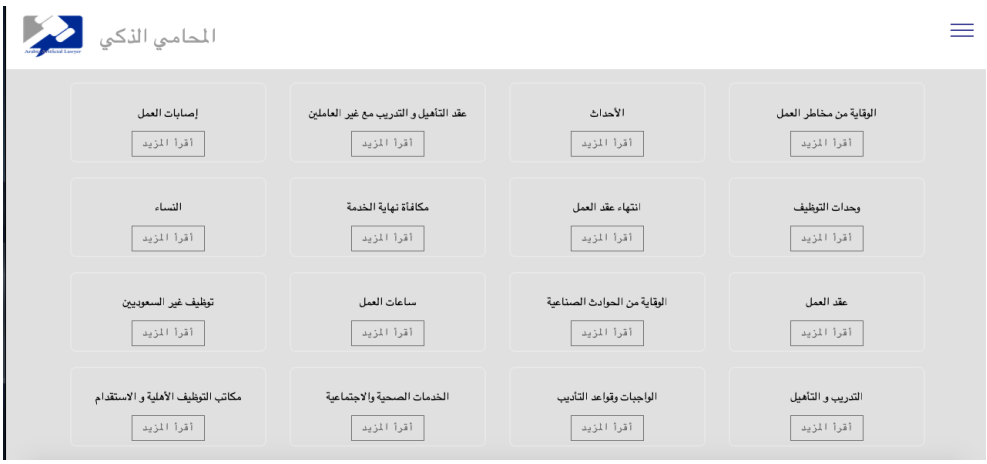


Figure 19: Regulation labor law list interface

- **Regulation labor law interface:** in this page the regulation of labor law appears as a text, so clients can read it and acknowledge their self, as shown in Figure 20 below.

Figure 20: Regulation labor law interface

- **Lawyer list interface:** this interface allows the client to view a list of volunteer lawyers they might communicate with in LiveChat, as shown in Figure 21 below.

Figure 21: Lawyer list interface

- **Chat with lawyer interface:** this interface allows the client to having a live chat with human lawyer and get his/her legal advice by using LiveChat [17] external application linked with our site, as shown in Figure 22 below.

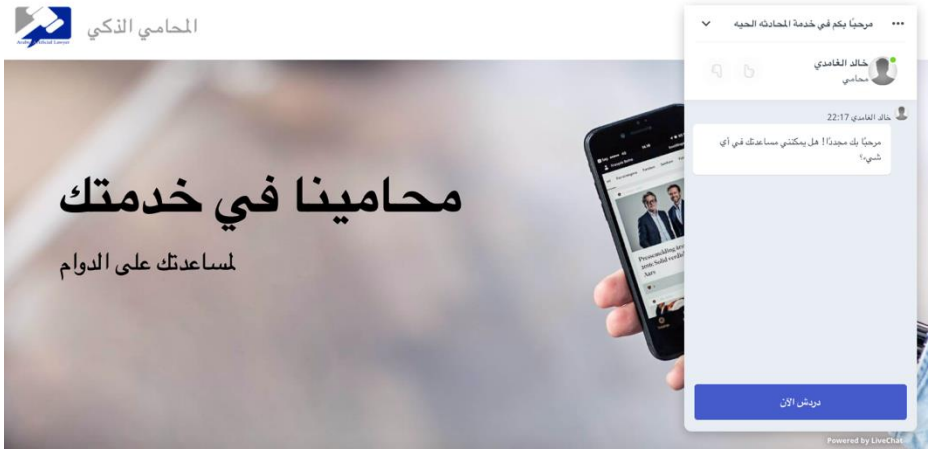


Figure 22: Chat with lawyer interface

- **Manual help page:** this page will view some information about us. As shown in Figure 23 below.

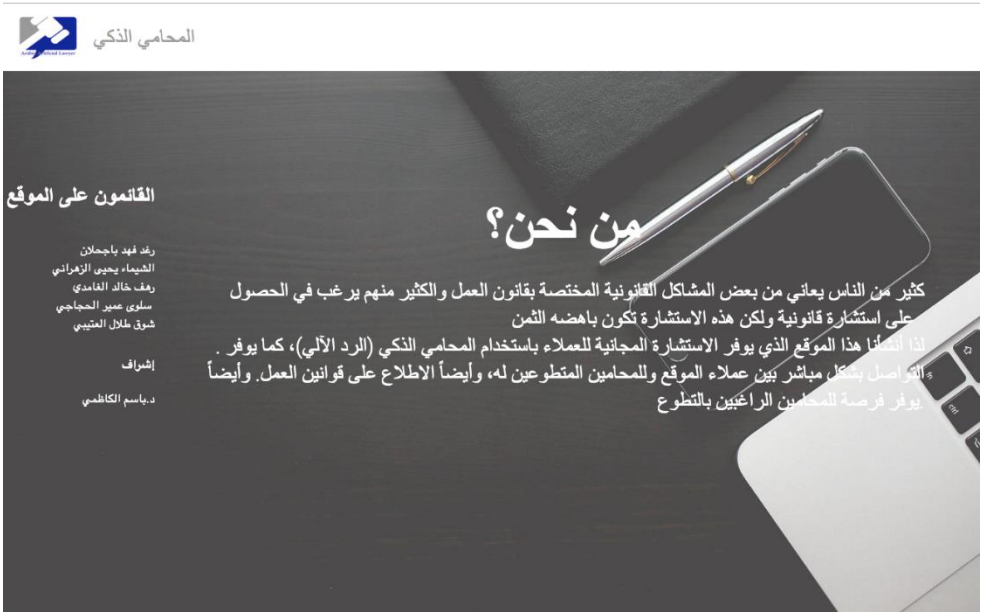


Figure 23: Manual help page interface

- **Manager view**
  - **Sign in page:** this interface shows the list of managers' services, as shown in Figure 24 below.

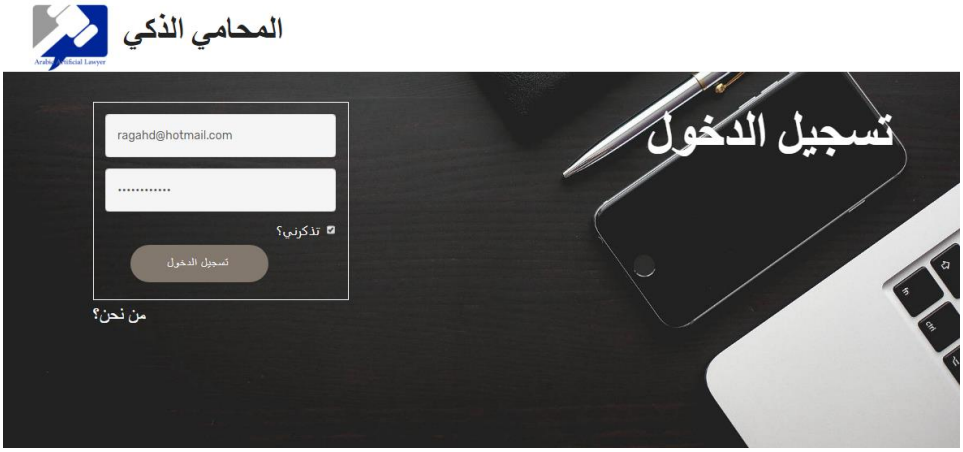


Figure 24: manger sign in interface

  - **Home page:** this interface shows the list of managers' services, as shown in Figure 25 below.

Figure 25: manger home page interface

- **Update profile interface:** this interface allows for manger to update the manager information, as shown in Figure 26 below.

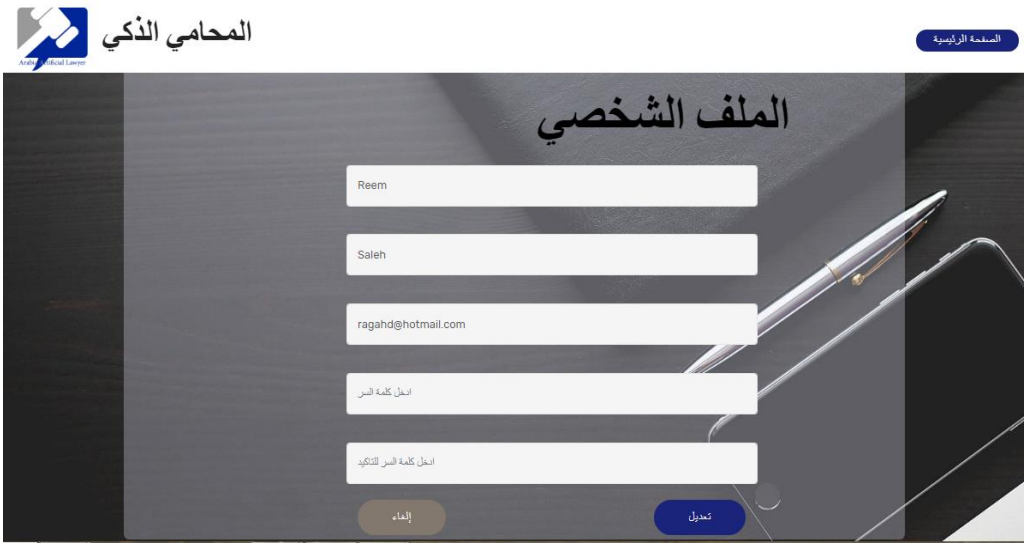


Figure 26: manger update profile interface

- **Lawyers forms interface:** this interface contains all forms that filled by the volunteer lawyers. There a buttons to show his/her information, as shown in Figure 27 below.

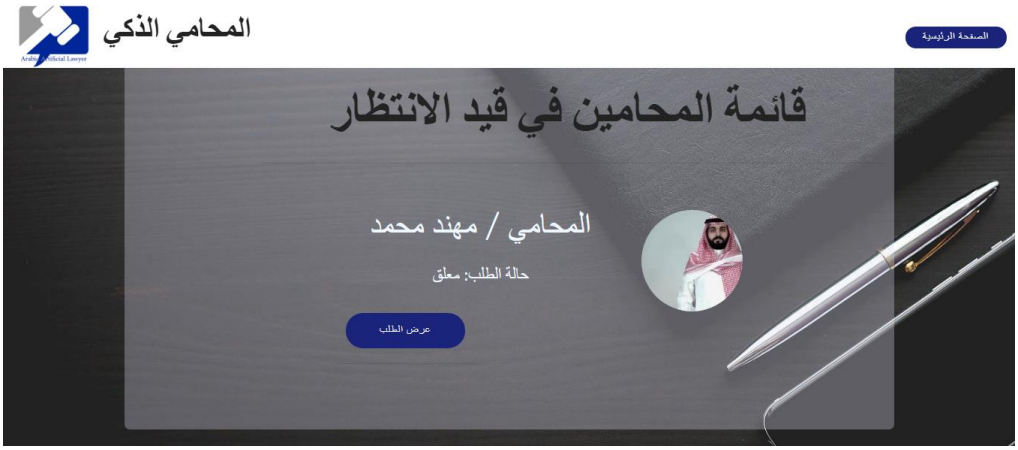


Figure 27: lawyer's forms interface

- **Lawyers list interface:** this interface shows the accepted lawyers' informations, name and other informations. There's a delete and view lawyer CV button as shown in Figure 28 below.

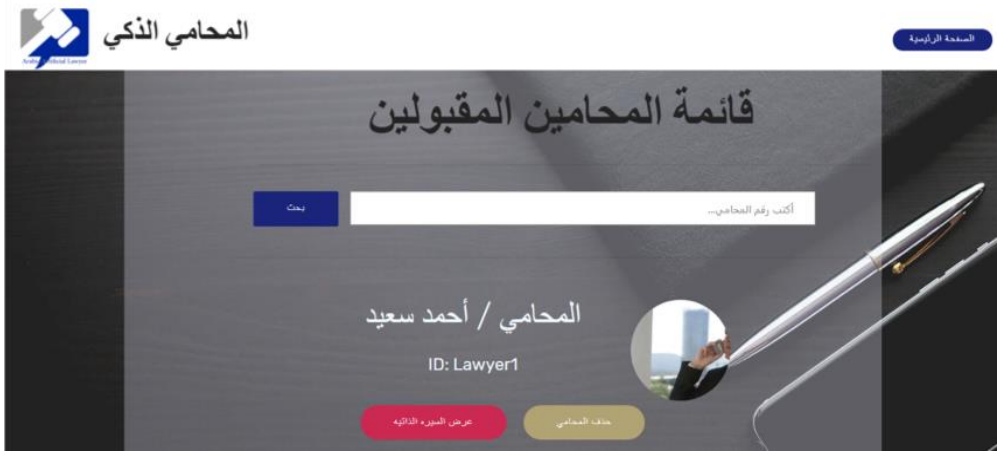


Figure 28: lawyers list interface

- **Help me page:** this page will view some information about as and email for connecting us. As shown in Figure 29 below.

Figure 29: help me interface

- **Lawyer view**
  - **Sign up by fill form interface:** in this interface the lawyer can join us as a volunteer lawyer by fill the form, as shown in Figure 30 below.

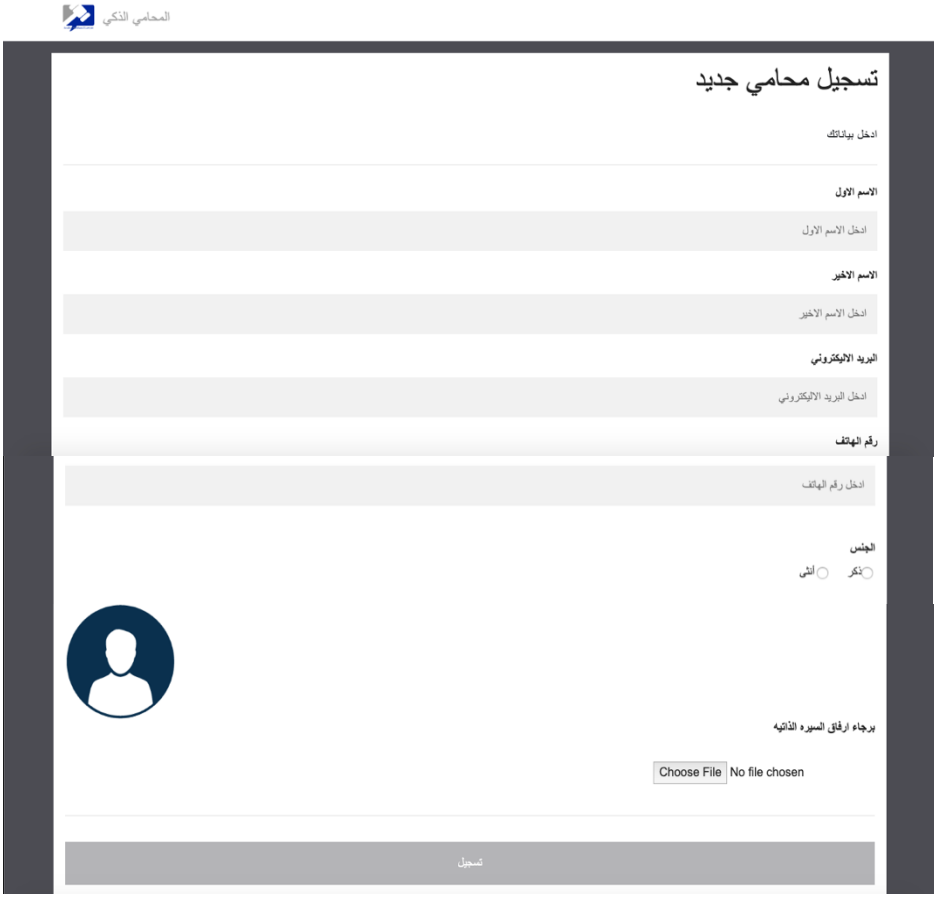


Figure 30: sign up by fill form interface

Chapter 5 IMPLEMENTATION

5. IMPLEMENTATION

This chapter describes the implementation of the AAL system.

5.1. AAL Chatbot agent

5.1.1. Data pre-processing

Our data sample (i.e Ministry of Labor and Development [13]) contains noisy data that needs to be omitted for the chatbot agent to work seamlessly. We utilized some of the freely available python packages to perform the necessary data cleaning.

- **Tools used in preprocessing stage are:**
  - **Natural Language Toolkit (NLTK):** this library is a heart of pre-processing step, which was explained previously in Chapter 1.
  - **Spyder:** is an IDE for Python programming that is equipped with all the necessary processing and analytical packages [23].

- **Pre-processing steps:**
  - **Organizing:** this is the first step to prepare our data for cleaning. The labor system documents are containing 24 categories for each category. So, we spilled each category into an individual text file and this file named by category's name which saved in uncleaned text directory. Then, we created a corresponding 24 empty CSV files has the same categories' names and saved it in cleaned CSV directory.
  - **Tokenizing:** using word_tokenize() function from the NLTK library so data spilled into words by a comma. As shown in Figure 31 below.
  - **Remove punctuation:** using maketrans() function from string library. As shown in Figure 31 below.
  - **Remove non-alphabetic tokens:** using isalpha() function to filter out the non-alphabetic tokens such as numbers. As shown in Figure 31 below.

- **Filter out stop words:** using stopwords.words('arabic') from nltk.corpus library to filter out all Arabic stop word such as 'الذين', 'إلا'. As shown in Figure 31 below.
- **Stemming**: using ISRIStemmer() function from NLTK library. As shown in Figure 31 below.
- **Writing cleaned data on csv file:** after all these steps the data will be cleaned enough to be written in a cleaned CSV file and saved into cleaned files directory using csv and io libraries. As shown in Figure 31 below.

```
# convert أ to ا and ة to ه
normlized_text=normalizeArabic(read_file)

# convert normlized_text sentences to spilt words that separated with comma
tokens = word_tokenize(normlized_text)

#remove punctuation from each word
remove_pun = str.maketrans('', '', string.punctuation)
stripped = [w.translate(remove_pun) for w in tokens]

# remove remaining tokens that are not alphabetic
patterns = [word for word in stripped if word.isalpha()]

#remove arabic stop stop
arabic_stop_word = stopwords.words('arabic')
stop_words = set(arabic_stop_word)
patterns = [w for w in patterns if not w in stop_words]

# stemming words which mean return the word to its root
stemer = ISRIStemmer()
stemmed = [stemer.stem(word) for word in patterns]

#for to write in the output CSV file
for cleaned_file in list_of_cleaned_files:
    #cheack tha fils have the same value => same name
    if(in_files == out_files):
        # chdir() function will used to change the path to (Cleaned CSV file) directory
        os.chdir(r'C:\\Users\\TOSHIBA\\Desktop\\python\\preproccessing\\Cleaned CSV file')
        #write it on ouptput file
        with io.open(cleaned_file, 'w', encoding='utf-8-sig') as out_file:
            writer = csv.writer(out_file)
            writer.writerow(stemmed)
        #closed the output CSV file
        out_file.close()
    #increments
    out_files = out_files + 1
in_files = in_files + 1
```

Figure 31: data cleaning code

By completing these steps, we have a Cleaned CSV files directory which contain 24 CSV files each one of them contain a list of words that related to its file name category. When each

one of these files' names will called **keywords** and the content of the files will know as **patterns**.

- **Responses:** as we divided the documents to 24 categories in pre-processing to create our keywords and patterns we created a 24 csv files with CSV UTF-8 format to save our responses on it. All response was taking from Ministry of Labor and Development [13]. Then we add Questiones.csv file which contain a more question to the client if artificial lawyer got confused about the response. As shown in Figure 32,33 below.
-

Questions	03/04/2019 10:35 ...	...ملف القيم المفصو	1 KB
إصابات العمل	03/04/2019 10:35 ...	...ملف القيم المفصو	6 KB
الإجازات	03/04/2019 10:35 ...	...ملف القيم المفصو	7 KB
الأجور	03/04/2019 10:35 ...	...ملف القيم المفصو	9 KB
الأحداث	03/04/2019 10:35 ...	...ملف القيم المفصو	5 KB
الأحكام العامة	03/04/2019 10:35 ...	...ملف القيم المفصو	12 KB
التدريب و التأهيل	03/04/2019 10:35 ...	...ملف القيم المفصو	3 KB
الخدمات الصحية والاجتماعية	03/04/2019 10:35 ...	...ملف القيم المفصو	5 KB
العمل البحري	03/04/2019 10:35 ...	...ملف القيم المفصو	9 KB
العمل الجزئي	03/04/2019 10:35 ...	...ملف القيم المفصو	1 KB
المناجم والمحاجر	03/04/2019 10:35 ...	...ملف القيم المفصو	6 KB
النساء	03/04/2019 10:35 ...	...ملف القيم المفصو	7 KB
الواجبات وقواعد التأديب	03/04/2019 10:35 ...	...ملف القيم المفصو	10 KB
الوقاية من الحوادث الصناعية	03/04/2019 10:35 ...	...ملف القيم المفصو	3 KB
الوقاية من مخاطر العمل	03/04/2019 10:35 ...	...ملف القيم المفصو	4 KB
انتهاء عقد العمل	03/04/2019 10:35 ...	...ملف القيم المفصو	11 KB
توظيف المعوقين	03/04/2019 10:35 ...	...ملف القيم المفصو	2 KB
توظيف غير السعوديين	03/04/2019 10:35 ...	...ملف القيم المفصو	6 KB
ساعات العمل	03/04/2019 10:35 ...	...ملف القيم المفصو	2 KB
عقد التأهيل و التدريب مع غير العاملين	03/04/2019 10:35 ...	...ملف القيم المفصو	4 KB
عقد العمل	03/04/2019 10:35 ...	...ملف القيم المفصو	7 KB
فترات الراحة والراحة الأسبوعية	03/04/2019 10:34 ...	...ملف القيم المفصو	6 KB
مكاتب التوظيف الأهلية و الاستقدام	03/04/2019 10:34 ...	...ملف القيم المفصو	2 KB
مكافأة نهاية الخدمة	03/04/2019 10:34 ...	...ملف القيم المفصو	3 KB
وحدات التوظيف	03/04/2019 10:34 ...	...ملف القيم المفصو	5 KB

Figure 32: Responses file.

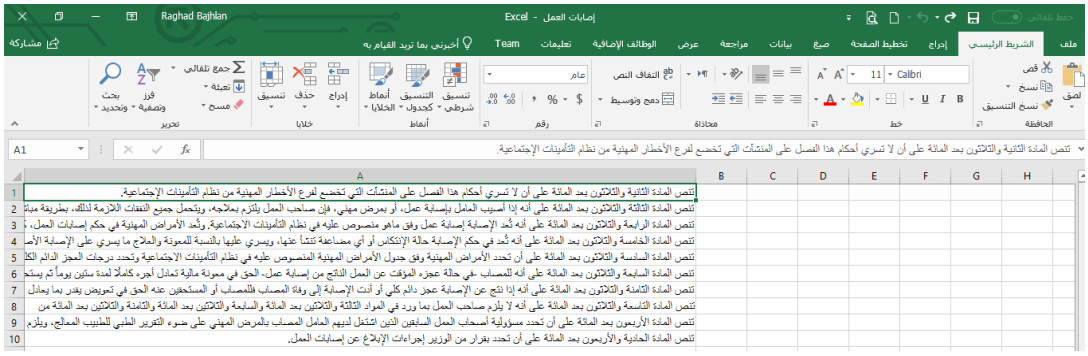


Figure 33: sample of response file.

5.1.2. AAL chatbot implementation

The hosting environment for AAL chatbot agent was implemented using the Jupyter Notebook framework [24] that provides interactive services for data processing and manipulation. we have implemented the Elizabith[18][19] algorithm for pattern matching where cleaned text can be matched to the available text snippets from the labor law regulations' data set.

5.1.2.1. Prepare data for matching

We started our implementation by built data structure to artificial lawyer chatbot in the following steps:

- **Keywords list:** a list that contain the 24 categories by reading files name in cleaned CSV directory and remove its .csv extension, as shown in Figure 34 below.

```
#create list to save csv files names
keywords=[]

#for loop to crete keywords list
for file in file_names:
    #remove files' extention
    file_name =file.split('.')[0]
    #add files' name to keywords list
    keywords.append(file_name)
#remove duplicate words
keywords = list(dict.fromkeys(keywords))
```

```
['إصابات العمل', 'الأجور', 'الأحداث', 'الأحكام العامة', 'الإجازات', 'التدريب
', 'و التأهيل', 'الخدمات الصحية والاجتماعية', 'العمل البحري', 'العمل الجزئي
' المناجم والمحاجر', 'النساء', 'الواجبات وقواعد التأديب', 'الوقاية من
الحوادث الصناعية', 'الوقاية من مخاطر العمل', 'انتهاء عقد العمل', 'توظيف
المعوقين', 'توظيف غير السعوديين', 'ساعات العمل', 'عقد التأهيل و التدريب
مع غير العاملين', 'عقد العمل', 'فترات الراحة والراحة الأسبوعية', 'مكاتب
التوظيف الأهلية و الاستقدام', 'مكافأة نهاية الخدمة', 'وحدات التوظيف']
```

Figure 34: Keywords list.

- **Cleaned_keywords_Dic dictionary:** in this step, our Cleaned_keywords_Dic code identified the normal form of a given phrase in terms of verbs and nouns as appropriate. An excerpt from the generated data is given in Figure below 35.

```
#create directory to save keywords after cleaning for matching
cleaned_keywords_Dic={}

#for loop to clean each keyword in keywords list
for key in keywords:
    #spilt the keyword name by comma
    tokens = word_tokenize(key)
    #remove punctuation from each word
    remove_pun = str.maketrans('', '', string.punctuation)
    words = [w.translate(remove_pun) for w in tokens]

    #remove non-alphabetic characters
    alphabetic_words = [word for word in words if word.isalpha()]

    #remove arabic stop stop
    arabic_stop_word = stopwords.words('arabic')
    stop_words = set(arabic_stop_word)
    alphabetic_words = [word for word in alphabetic_words if not word in stop_words]

    #stem each word
    stemer = ISRIStemmer()
    stemmed_words = [stemer.stem(word)  for word in alphabetic_words]
    cleaned_keywords_Dic[key] = stemmed_words

#foor loop to remove "عمل" word
for key in cleaned_keywords_Dic:
    values = cleaned_keywords_Dic[key]
    #giving index for each value in vlaues list
    for i,v in enumerate(values):
        if v == 'عمل':
            #delete the word
            del values[i]
```

```
إصابات العمل': ['اصب'], 'الأجور': ['اجر'], 'الأحداث': ['حدث'], 'الأحكام'}
'] ': 'العامة': ['حكم', 'عمة'], 'الإجازات': ['اجز'], 'التدريب و التأهيل
' ,'درب', 'و', 'أهل'], 'الخدمات الصحية والاجتماعية': ['خدم', 'صحة
اجتماعية'], 'العمل البحري': ['بحر'], 'العمل الجزئي': ['جزئ'], 'المناجم
':والمحاجر': ['نجم', 'حجر'], 'النساء': ['نسء'], 'الواجبات وقواعد التأديب
' ,'['وجب', 'قعد', 'أدب'], 'الوقاية من الحوادث الصناعية': ['وقي', 'حدث
'] ': 'صنع'], 'الوقاية من مخاطر العمل': ['وقي', 'خطر'], 'انتهاء عقد العمل
':'نهء', 'عقد'], 'توظيف المعوقين': ['وظف', 'عوق'], 'توظيف غير السعوديين
'وظف', 'سعد'], 'ساعات العمل': ['ساع'], 'عقد التأهيل و التدريب مع غير
العاملين': ['عقد', 'أهل', 'و', 'درب'], 'عقد العمل': ['عقد'], 'فترات
الراحة والراحة الأسبوعية': ['فتر', 'رحة', 'رحة', 'سبع'], 'مكاتب التوظيف
الأهلية و الاستقدام': ['كتب', 'وظف', 'هلي', 'و', 'قدم'], 'مكافأة نهاية
الخدمة': ['كفأ', 'نهي', 'خدم'], 'وحدات التوظيف': ['وحد', 'وظف'}
```

Figure 35: Cleaned_keywords_Dic dictionary.

- **Keywords_patterns_dic dictionary:** Is the data dictionary that was generated after data cleaning in the previous step. It describes the correlation

between a given term and their corresponding matches from the sample data set (i.e. النساء). As shown in Figure 36 below.

```
#create a list to save each csv file contents
temp_pattern_list=[]
#create a directory to link each keyword with its patterns
keywords_patterns_dic={}
#for loop to linking each keyword with its patterns
for key in keywords: #taking the keywords
    for pattern_file in file_names: #taking the patterns
        os.chdir(cleaned_file_path)
        with open(pattern_file, encoding='utf-8-sig') as csv_file:
            #pattern list of each file
            patterns_content = csv.reader(csv_file, delimiter=',')
            #remove files' extension
            file_name = pattern_file.split('.')[0]

            #matching each file name with each key in keyword
            if( key == file_name):
                for pattern in patterns_content:
                    #adding patterns in temp_pattern_list
                    temp_pattern_list.extend(pattern)
                    #remove duplicate patterns
                    temp_pattern_list = list(dict.fromkeys(temp_pattern_list))

                #adding keywoeds with its patterns to keywords_patterns_dic dictionary
                keywords_patterns_dic[key] = temp_pattern_list
                #clean the temprory list to repeat the matching step for each file
                temp_pattern_list=[]
```

```
'إصابات العمل': ['تسر', 'حكم', 'فصل', 'منش', 'خضع', 'فرع', 'خطر', 'هنه', 'نظم',
'امن', 'اجتماعيه', 'اذا', 'اصب', 'عمل', 'او', 'مرض', 'فان', 'صحب', 'لزم', 'حمل',
'جمع', 'نفق', 'طرق', 'بشر', 'قمه', 'فحص', 'حلل', 'جهز', 'نقل', 'الى', 'اما',
'علج', 'تعد', 'صبه', 'وفق', 'اهو', 'نصص', 'يعد', 'ارخ', 'اول', 'شهد', 'طبه',
'حله', 'نكس', 'اي', 'ضعف', 'تنش', 'يسر', 'عليها', 'نسب', 'عون', 'اصل', 'حدد',
'جدل', 'درج', 'عجز', 'دئم', 'كلي', 'جزئ', 'دلل', 'ذكر', 'صاب', 'ؤقت', 'نتج',
'لحق', 'اله', 'عدل', 'اجر', 'كمل', 'لمد', 'ستن', 'يوم', 'سحق', 'قبل', 'مال',
'طول', 'مده', 'غرق', 'فذا', 'بلغ', 'سنه', 'قرر', 'طبا', 'عدم', 'شفه', 'وحل',
'صحه', 'تمك', 'عدت', 'نهى', 'عقد', 'يعض', 'يكون', 'حق', 'ردد', 'دفع', 'خلل',
'ادت', 'وفه', 'عنه', 'عوض', 'قدر', 'ثلث', 'سنو', 'بحد', 'دنى', 'ربع', 'خمس',
'الف', 'ريل', 'ضرب', 'قيم', 'ورد', 'واد', 'ائه', 'سبع', 'ثمن', 'ثبت', 'يتي',
'ان', 'عمد', 'نقص', 'حدث', 'سبب', 'سوء', 'سلك', 'قصد', 'جنب', 'منع', 'عرض',
'طبب', 'كلف', 'شرع', 'سؤل', 'سبق', 'شغل', 'لدي', 'هني', 'ضوء', 'طبي', 'اده',
'قضا', 'شرط', 'تكون', 'صنع', 'مهن', 'ارسو', 'ينش', 'وزر', 'جرء']
```

Figure 36: Keywords_patterns_dic dictionary.

- **Creating matching_counter_dic dictionary:** a (key, value) pair is generated for each word or phrase where the key is the word/phrase and the

value is the count of its occurrences. An excerpt is given in Figure 37.

```
#Create a dictionary for counting the matching
matching_counter_dic ={}
for key in keywords:
    matching_counter_dic[key]=0
```

```
{'إصابات العمل': 0, 'الأجور': 0, 'الأحداث': 0, 'الأحكام العامة': 0, 'الإجازات':
'التدريب و التأهيل': 0, 'الخدمات الصحية والاجتماعية': 0, 'العمل البحري': 0,
العمل الجزئي': 0, 'المناجم والمحاجر': 0, 'النساء': 0, 'الواجبات وقواعد': 0,
التأديب': 0, 'الوقاية من الحوادث الصناعية': 0, 'الوقاية من مخاطر العمل': 0,
' ,انتهاء عقد العمل': 0, 'توظيف المعوقين': 0, 'توظيف غير السعوديين': 0'
'ساعات العمل': 0, 'عقد التأهيل و التدريب مع غير العاملين': 0, 'عقد العمل
0, 'فترات الراحة والراحة الأسبوعية': 0, 'مكاتب التوظيف الأهلية و الاستقدام':
0, 'مكافأة نهاية الخدمة': 0, 'وحدات التوظيف': 0}
```

Figure 37: matching_counter_dic dictionary.

5.1.2.2. Artificial lawyer pattern matching

In this part we explained how we implemented the Elizabith[18][19] pattern matching algorithm with some additional steps to make it close to our chatbot model.

- **Creating replacement dictionary:** this dictionary will replace a popular word that may use by clients to words that used by Ministry of Labor and Development document [13]. As shown in Figure 38 below.

```
replacments = {
    "مسئول": "صاحب العمل", "مسؤول": "صاحب العمل", "مدير": "صاحب العمل",
    "وزارة": "وزارة العمل", "راتب": "أجر", "إختبار": "تجربة", "المرأة": "النساء",
    "إجازته": "إجازة", "الأجانب": "الغير سعوديين", "الاطفال": "الاحداث"
}
```

Figure 38: replacement dictionary.

- **Clean_up_sentence function:** is a function used to clean users input

which takes one argument as string and cleaned it by using pre-processing steps (tokenizing, removing punctuations, non-aliphatic characters, and Arabic stop words, and stemming the sentence). The output of this function is a list of cleaned words. As show in Figure 39 below.

```
#clean up user's input
def clean_up_sentence(sentence):
    normalizeArabic(sentence)
    #using replacment dictionary to replace a popular words that clients may use
    for old,new in replacments.items():
        sentence=sentence.replace(old, new)
    sentence=sentence.replace('؟', ' ')
    #tokenize the pattern
    tokens = word_tokenize(sentence)
    #remove punctuation from each word
    remove_pun = str.maketrans('', '', string.punctuation)
    words = [w.translate(remove_pun) for w in tokens]
    #remove non-alphabetic characters
    alphabetic_words = [word for word in words if word.isalpha()]
    #remove arabic stop stop
    arabic_stop_word = stopwords.words('arabic')
    stop_words = set(arabic_stop_word)
    alphabetic_words = [word for word in alphabetic_words if not word in stop_words]
    #stem each word
    stemer = ISRIStemmer()
    stemmed_words = [stemer.stem(word) for word in alphabetic_words]
    stemmed_words = list(dict.fromkeys(stemmed_words))
    return stemmed_words
```

Figure 39: clean_up_sentance function.

- **Matching function:** this function takes user input and clean it using clean_up_sentance function and then match words of the output list with:
    - The values of cleaned_keywords_Dic dictionary.
    - The value of keywords_patterns_dic dictionary.

Each matching means increasing in matching_counter_dic dictionary value by 1 to key that matched. As shown in Figure 40 below.

```
#matching between client's query and cleaned_keywords_Dic dictionary
#matching between client's query and keywords_patterns_dic dictionary
def matching(sentence):
    #clean up client's input
    client_query = clean_up_sentence(sentence)
    matching_counter_dic={}
    #matching client's query with cleaned_keywords_Dic dictionary
    for client_word in client_query:
        #matching each client's word with keywords
        for keywords,cleand_key_list in cleaned_keywords_Dic.items():
            #matching client's word with each cleaned key
            for key in cleand_key_list:
                #if there is a match then increase value of matching_counter_dic key by 1
                if client_word == key:
                    matching_counter_dic[keywords] = matching_counter_dic.get(keywords, 0) + 1
                else:
                    matching_counter_dic[keywords] = matching_counter_dic.get(keywords, 0)

        #matching client's query with patterns and count each keyword how much appear in client'
        for key,pattern_list in keywords_patterns_dic.items():
            for pattern in pattern_list:
                if client_word == pattern:
                    matching_counter_dic[key] = matching_counter_dic.get(key, 0) + 1
                else:
                    matching_counter_dic[key] = matching_counter_dic.get(key, 0)
    return matching_counter_dic
```

Figure 40: matching function

- **Match_filtering function:** in this function take a single argument which is the client's input and called the matching function and send this argument to it. the matching will match the sentence and return matching_counter_dic dictionary. Then it will be comparing the value of each key to find out the highest match as the best match and return its key which contains the keyword. But if we have more than 3 keys have the same value and this value is the highest match then this function will return a Questions.csv file to ask more question to the client. As shown in Figure 41 below.

```
#to generate the keyword who has the highest counter number
def match_filtering(sentence):
    #dictionary for keywords and its number of repetition
    matching_result = matching(sentence)
    #best match will used to hold the highest counter value and it's initialize to 0
    best_matching=int()
    #list that contain keywords which have the highest counter
    key_matched=[]
    #search for best matching keywords
    for key,counter in matching_result.items():
        if counter >= best_matching:
            best_matching = counter
    #put the best matched keyword in key_matched list
    for key,counter in matching_result.items():
        if counter == best_matching:
            key_matched.append(key)
    #check if client's query didn't match any thing in our dataset or match greater than 3 keywords
    #then need specific input to get more appropriate keyword
    if((len(key_matched) == 0) or (len(key_matched) > 3 )):
        key_matched=[]
        key_matched.append('Questions')
    return key_matched
```

Figure 41: matching filtering function.

- **Make_response_dic function:** this function takes a single argument as a string which contains the client's input then called the match_filtering function and send this argument to it, the match_filtering function will return the keyword of best matching. Then, it uploads all responses in the response directory that have the same name of keyword and placed it in the response_Dic dictionary when each response is a key and all the values assigned to zero. As shown in Figure 42 below.

```
def make_response_dic(sentence):
    #get the best matching keywords
    match_filter = match_filtering(sentence)
    response_files=[]
    response_Dic ={}
    #change directory to responses folder
    os.chdir(responsesFolderPath)
    #put response .csv files name for the best matching keywords in response_files list
    for key in match_filter:
        response_files.append(key+".csv")
    #open each file in 'utf-8-sig' format
    for file in response_files:
        with open(file, encoding='utf-8-sig',errors='replace') as csv_file:
            # read all responses for the best matching keywords to get responses list
            responses = csv.reader(csv_file)
            if file == 'Questions.csv':
                #read each lines separately
                lines = open(file, encoding='utf-8-sig',errors='replace').read().splitlines()
                #choose random line for response
                row = random.choice(lines)
                response_Dic[row]=0
            else:
                #separate each response in seperate row from responses list
                for response in responses:
                    #save response as a key in response_Dic dictionary and assign its value t
                    for row in response:
                        response_Dic[row]=0
    return response_Dic
```

Figure 42: Make_response_dic function.

- **Response function:** this function takes a single argument as a string which is the client's input and called the make_response_dic function and send the argument to it, the make_response_dic will return a response_Dic. Then, the function matching cleaned client's input with each response in the dictionary then filtering out the best match and return it to the client. As shown in Figure 43 below.

```python
def response(sentence):
    #for greeting query
    greet= normalizeArabic(sentence)
    #replying by appropriate greet
    for key,val in greeting_dic.items():
        if(greet==key):
            return val

    #clean up client's query
    client_query=clean_up_sentence(sentence)
    #get all responses of the best matching keywords
    responses=make_response_dic(sentence)

    for response,counter in responses.items():
        #clean up the responses
        cleaned_response=clean_up_sentence(response)
        #matching each word in client's input with cleaned response and
        #count each respons how much appear its words in client's input
        for client_word in client_query:
            for cleand_word in cleaned_response:
                if client_word == cleand_word:
                    responses[response] = responses.get(response, 0) + 1
                else:
                    responses[response] = responses.get(response, 0)
    best_matching=int()
    responses_matched=[]
    #find the best matching response
    for response,counter in responses.items():
        if counter >= best_matching:
            best_matching = counter
    for response,counter in responses.items():
        if counter == best_matching:
            responses_matched.append(response)
    return responses_matched[0]
```

Figure 43: response function.

5.2. AAL website

We built our website using PHP, HTML, CSS, and JavaScript as covered in Chapter 4.

5.2.1. LiveChat Connection

We have included the LiveChat[17] on the Lawyer list page that shows the lawyers might you will be getting responses from them. As shown in Figure 44 there is a chat icon in the right-bottom side of the screen.

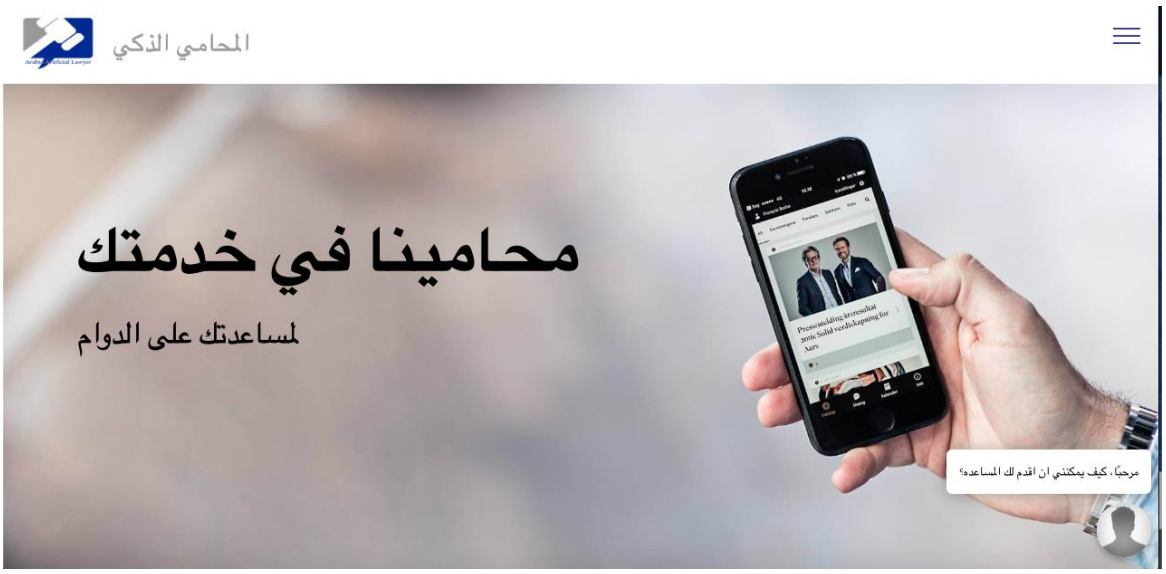


Figure 44: LiveChat.

5.2. Linking chatbot to AAL web-based system

Used flask library to link the Artificial lawyer to AAL hosting system which consider a microframework for Python based on Werkzeug and Jinja2 [25].

- **HTML section:**
  - Normally flask doesn't convenient with HTML tags. So, we built our HTML using Jinja2 blocks instead of HTML tags [25].
  - Used url_for() function. To build a URL to a specific function [25], which use to implement the CSS, JavaScript, and Images.
- **Python section:**
  - Used the route() decorator to bind a function to a URL[25]. As shown in Figure 45 below.
  - Used render_template() function to generating HTML from within python[25]. As shown in Figure 45 below.
  - Used jsonify() function which turns the JSON output into a response without reload the page[25]. As shown in Figure 45 below.

```
app = Flask(__name__)


@app.route('/')
def index():
    return render_template('indexme.html')

@app.route('/background_process')
def background_process():
    lang = request.args.get('user_input')
    lang = str(lang)
    result = response(lang)
    result = str(result)
    # dict[lang] = result
    return jsonify(user_input=lang, response=result)

if __name__=='__main__':
    app.run(debug=True,port=5000)
```

Figure 45: Flask code.

5.3. Deploying AAL system

We deployed AAL system into two servers then linked it using URL, AAL chatbot deployed in Heroku server [27] which can handle with Python and Flask library and AAL web-based system in byethost.com [26] which can handle HTML and PHP languages.

5.3.1. AAL chatbot deployment

Heroku is a cloud platform that lets companies build, deliver, monitor [27]. Before deploying the chatbot into Heroku we need to use github and Gunicor library is need for some steps in the code:

- Change the path of data using os library and dirname() function to specific the directory name and join() function to specific the data files. As shown in Figure 46 below.

```
here = os.path.dirname(os.path.abspath(__file__))
# path to laod Artificial lawyer data structer
pickleDataPath = os.path.join(here, "ArtificialLawyerDataStructure")
# path to acess to responses directory
responsesFolderPath = os.path.join(here, "Response")
```

Figure 46: redeclare files path for Heruko server

- Initialized a repository using **git init** command [27].
- Create app using a **heroku create -a aalchatbot** command to create https://aalchatbot.herokuapp.com/ URL [27].
- Create a requerments.txt file using **pip freeze > requerments.txt** command, which contain requirement library for chatbot [27]. As shown in Figure 47 below.

requirements.txt

```
Flask==1.0.2
Flask-Cors==3.0.7
nltk==3.4
gunicorn==19.9.0
```

Figure 47: requirments.txt file

- Create a Procfile which will contain a system type which in our case a web, and script name which in our case called main.py in using Gunicor library [27]. As shown in Figure 48 below.

Procfile

```
web: gunicorn main:app --log-file -
```

Figure 48: Procfile file

- Add files to app using **git add .** and **git commit -m "done"** commands [27].
- Finally, deployed AAL chatbot system using **git push heroku master** command [27]. Then it deployed at create

https://aalchatbot.herokuapp.com/, As shown in Figure 49 below.

Figure 49: deployment of AAL chatbot agent.

5.3.2. Web-based system deployment

We upload our websites files on byethost.com [26] free hosting server. Our reserved domain is aalawyer.byethost12.com. Then create a live Database on the same server for accessing and manipulating the data.

Chapter 6 VALIDATION

5. VALIDATION

This chapter describes the testing of AAL system using information retrieval chatbot it provides testing and training of the system.

6.1 Validation

6.1.1 Artificial lawyer evaluation

To evaluate the chatbot agent's functionality, we have gathered a number of 99 questions that are believed commonly asked questions by people listed in the FAQ section of the ministry web site [28]. We followed similar evaluation approach as that was used for the Quran's chatbot agent evaluation [29]. Their approach categorized chatbot responses as: related, partially related, and not related as follows:

- Related: in case the answer was exactly matching the context of the provided question's topic and is correct.
- Partially related: in case the answer was not semantically correct but falls within in the same topic context.
- Not related: in case the answer was not correct and falls in a different topic.

we wrote a code for evaluation thus each query got a response from chatbot agent then we evaluated if the response is related or partially related or not related at all (when value 1 mean Related, 2 Partially Related, and 3 Not Related). As shown in Figure 50 below.

```
evaluation_dic ={"Related":0, "Partially Related":0, "Not Related":0}
def evaluation():
#     evaluation_dic ={"Related":0, "Partially Related":0, "Not Related":0}
    os.chdir('/Users/salwaalhajjaji/Desktop')
    file="valid_questions.csv"
    with open(file, encoding='utf-8-sig') as csv_file:
        #valid q. list of one file
        questions_list = open(file, encoding='utf-8-sig').read().splitlines()
    for question in questions_list:
        print("Question "+question)
        response_list = response(question)
        print(response_list)
        input_user = input("Type 1 if it's Related, 2 if it's Partially Related, 3 if it's Not Related")
        if input_user=="1":
            evaluation_dic["Related"] = evaluation_dic.get("Related", 0) + 1
        elif input_user=="2":
            evaluation_dic["Partially Related"]= evaluation_dic.get("Partially Related", 0) + 1
        elif input_user=="3":
            evaluation_dic["Not Related"]= evaluation_dic.get("Not Related", 0) + 1
```

Figure 50: evaluation code.

- First evaluation in Figure 51 below was done for first version of code when no remove duplicates or normalization was using.

Classfication	Precentg	
Rleated	30	0
Parttaliy Related	9	1
Not Related	60	2

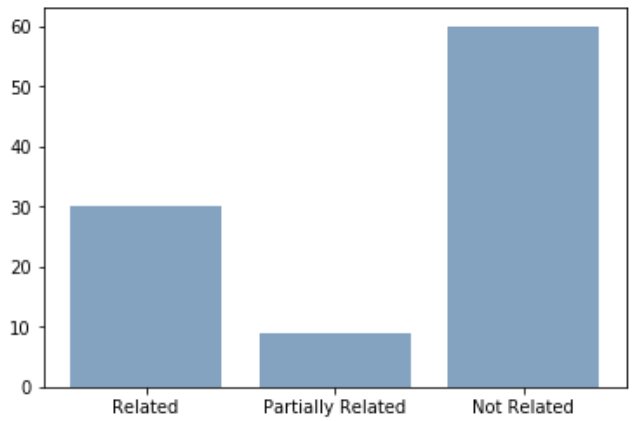


Figure 51: first evaluation.

This version of artificial lawyer is not considered because the low rate.

- Second evaluation in Figure 52 below done to second version of the code when remove duplicates was adding to clean_up_sentence (sentence) function to remove duplicate from user query.

Classfication	Precentg	
Rleated	39	0
Parttaliy Related	20	1
Not Related	40	2

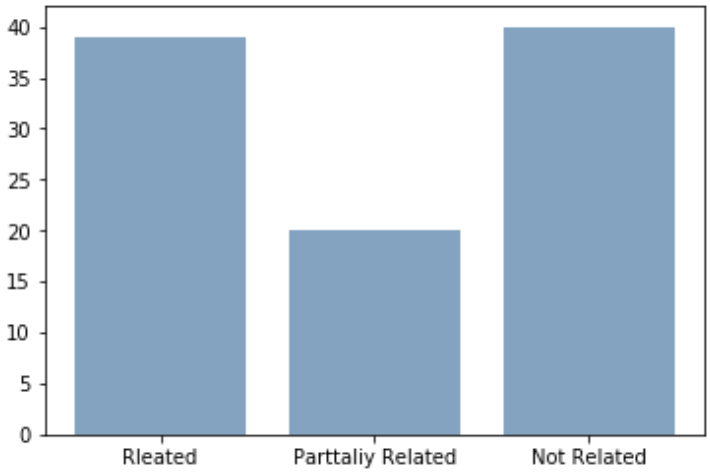


Figure 52: second evaluation.

The addition in this version of artificial lawyer is considered because its improved the rate.

- Third evaluation in Figure 53 below has been done for third version of the code when normalization was adding to cleaning data and to clean_up_sentence(sentence) function using **re** library which indicates:
    - To remove Arabic language symbols like (ٰ,ٓ,ٔ,ٖ,ٚ,ٛ).
    - Return any (ٱ,آ,إ) to (ا).
    - Return any (ة) to (ۃ).

Classfication	Precentg	
Rleated	49	**0**
Parttaliy Related	11	**1**
Not Related	39	**2**

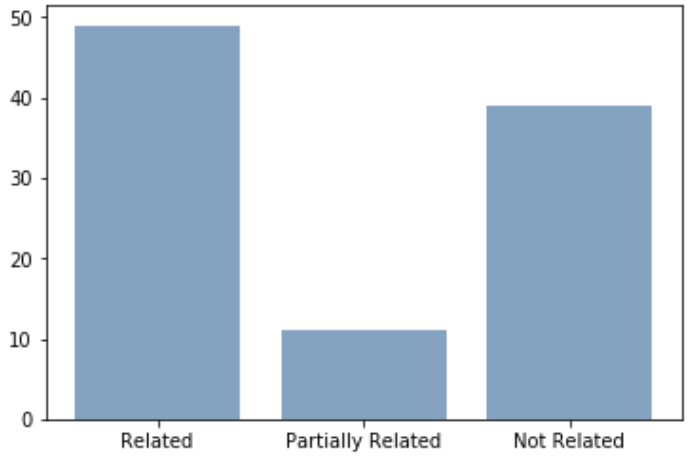


Figure 53: Third evaluation.

The addition in this version of artificial lawyer is considered because its improved the rate.

- Fourth evaluation in Figure 54 below was done by replace any management position word in cleaning data and clean_up_sentence(sentence) to (صاحب عمل).

Classfication	Precentg	
Rleated	45	0
Parttaliy Related	5	1
Not Related	49	2

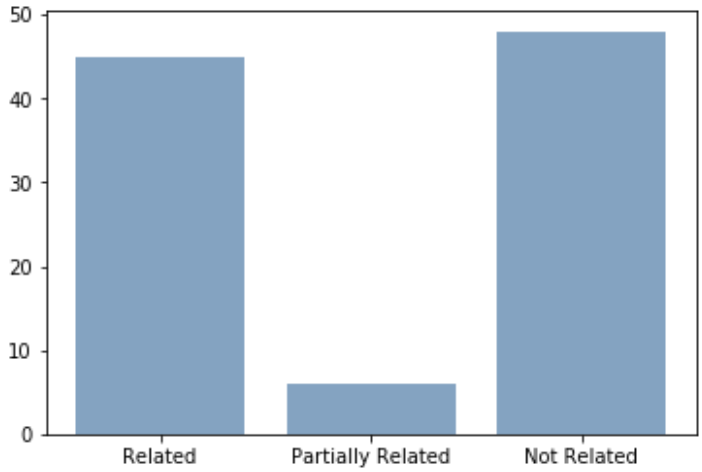


Figure 54: fourth evaluation.

This version of artificial lawyer is not considered because the low rate.

At the end we used the third version of code as official version.

6.1.2 AAL hosting webpage validation

- **Lawyer**
  - **Volunteerism form page:** in this interface we check the information validation that's the lawyer provided, as shown in Figure 55 below and show the updated database in Figures 56, 57.

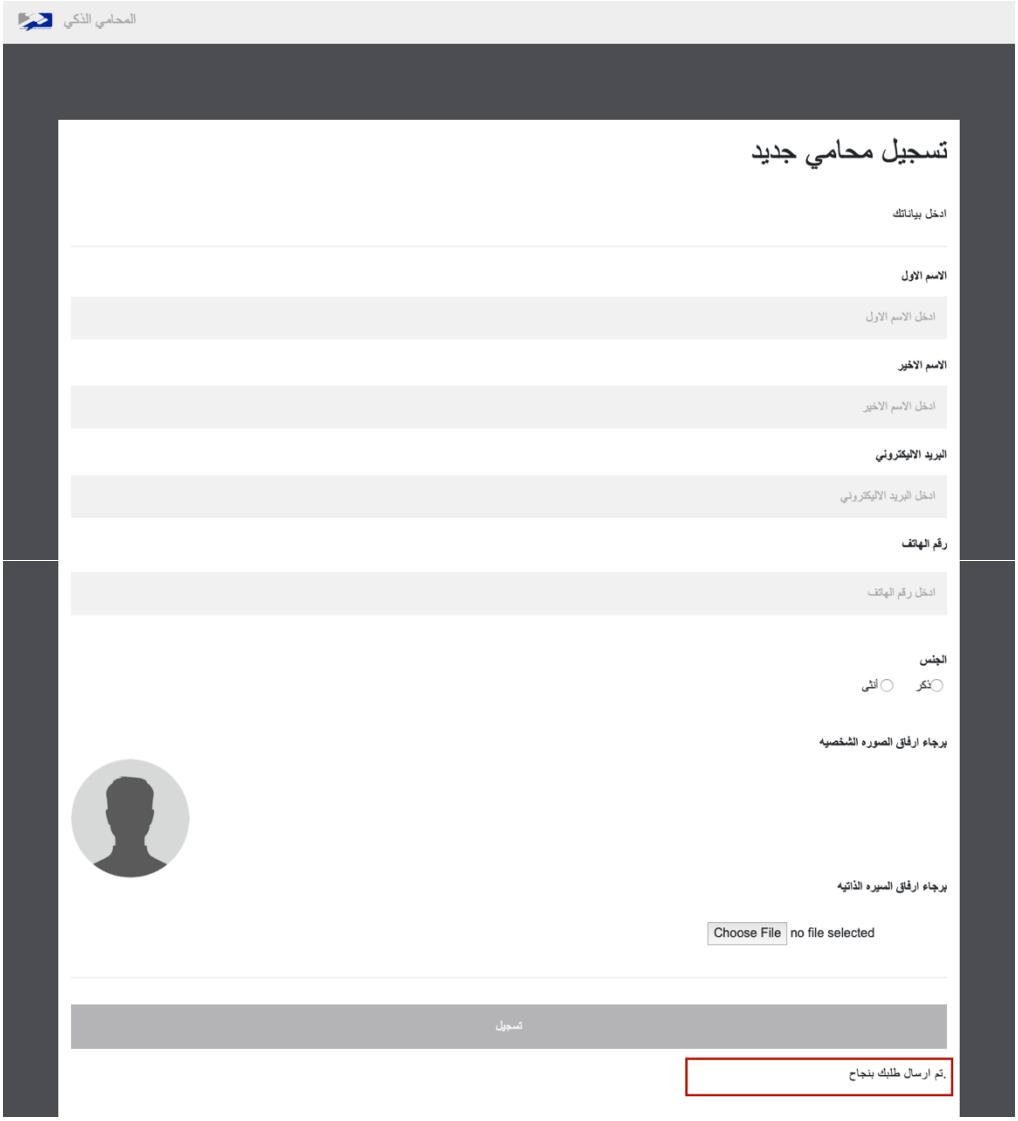
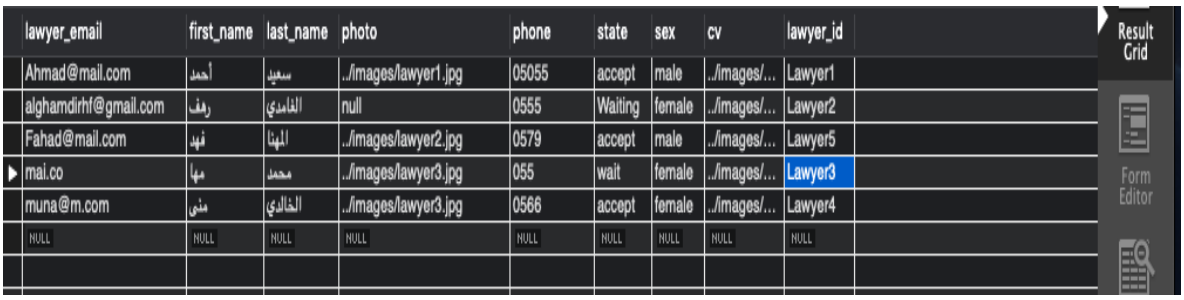


Figure 55: Lawyer volunteerism form interface.



lawyer_email	first_name	last_name	photo	phone	state	sex	cv	lawyer_id
Ahmad@mail.com	أحمد	سعيد	../images/lawyer1.jpg	05055	accept	male	../images/...	Lawyer1
alghamdirhf@gmail.com	رهف	الغامدي	null	0555	Waiting	female	../images/...	Lawyer2
Fahad@mail.com	فهد	المهنا	../images/lawyer2.jpg	0579	accept	male	../images/...	Lawyer5
mai.co	مها	محمد	../images/lawyer3.jpg	055	wait	female	../images/...	Lawyer3
muna@m.com	منى	الخالدي	../images/lawyer3.jpg	0566	accept	female	../images/...	Lawyer4
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 56: Database for insert lawyer information.

lawyer_email	first_name	last_name	photo	phone	state	sex	cv	lawyer_id
Ahmad@mail.com	أحمد	سعيد	../images/lawyer1.jpg	05055	accept	male	../images/...	Lawyer1
alghamdirhf@gmail.com	رهف	الغامدي	null	055555555	Waiting	female	cv.pdf	0
alghamdirhf@gmail.com	رهف	الغامدي	null	0555	Waiting	female	../images/...	Lawyer2
Fahad@mail.com	فهد	المهنا	../images/lawyer2.jpg	0579	accept	male	../images/...	Lawyer5
mai.co	مها	محمد	../images/lawyer3.jpg	055	wait	female	../images/...	Lawyer3
muna@m.com	منى	الخالدي	../images/lawyer3.jpg	0566	accept	female	../images/...	Lawyer4
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 57: Database for inserted lawyer information.

- **Manager**

  - **Sign in page:** this interface allows for manger to sign in to his/her account and check the information validation, if it's not valid will print out this message as shown in Figure 58 below.

Figure 58: manager sign in validation.

- **Edit profile page:** this interface allows for manger to update his/her information, as shown in Figures 59, 60, 61 below.

manager_id	manager_email	password	first_name	last_name
m_123	ragahd@hotmail.com	716b1d667599c2f30fd9a7d04068c6ad15cea38...	Raghad	Bajhlan
m_234	alshaimaa_@hotmail.com	e2520e99f3f3134d4c8b5ed724b2190e3f6dcb6...	Alshimaa	Alzhrani
m_267	salwa.alhajjaji@hotmail.com	ea403503b01aebebc1682f91b8379f66fcbdbe3...	Salwa	Omea
NULL	NULL	NULL	NULL	NULL

Figure 59: Database before edit manager's profile.

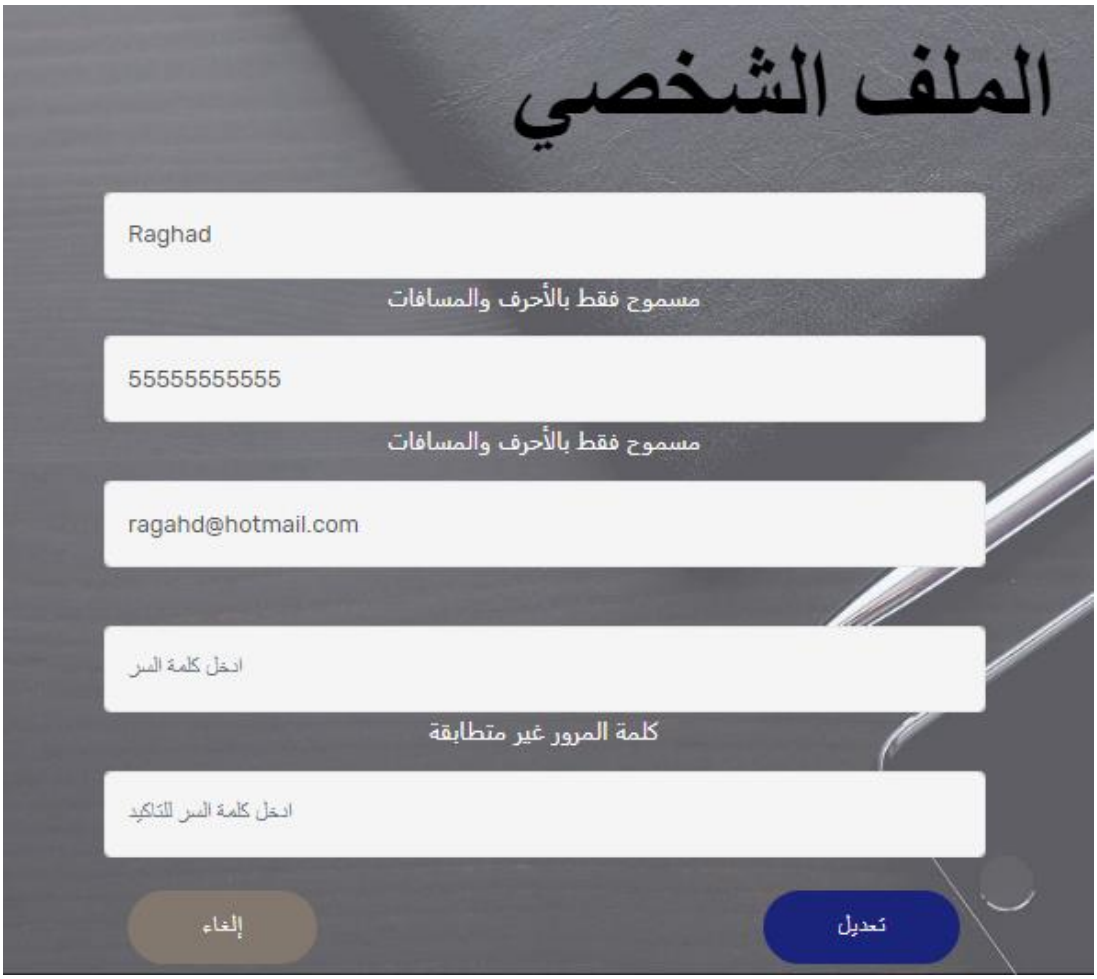


Figure 60: Then deletion information messages will appear to the manager

manager_id	manager_email	password	first_name	last_name
m_123	ragahd@hotmail.com	716b1d667599c2f30fd9a7d04068c6ad15cea38...	Reem	Saleh
m_234	alshaimaa_@hotmail.com	e2520e99f3f3134d4c8b5ed724b2190e3f6dcb6...	Alshimaa	Alzhrani
m_267	salwa.alhajjaji@hotmail.com	ea403503b01aebebc1682f91b8379f66fcbdbe3...	Salwa	Omea
NULL	NULL	NULL	NULL	NULL

Figure 61: Database after change manager's name

- **Delete lawyer:** this interface allows for manger to delete lawyer information, as shown in Figures 62, 63, 64, 65 below.

lawyer_email	first_name	last_name	photo	phone	state	sex	cv	lawyer_id
Ahmad@mail.com	أحمد	سعيد	../images/lawyer1.jpg	05055	accept	male	../images/Education2.jpg	Lawyer1
Fahad@mail.com	فهد	المهنا	../images/lawyer2.jpg	0579	accept	male	../images/AlShaimaaYahya.pdf	Lawyer5
mai.co	مها	محمد	../images/lawyer3.jpg	055	wait	female	../images/AlShaimaaYahya.pdf	Lawyer3
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 62: Database before deleting lawyer

Figure 63: Deletion confirmation messages

Figure 64: Then deletion information messages will appear to the manager

lawyer_email	first_name	last_name	photo	phone	state	sex	cv	lawyer_id
Fahad@mail.com	فهد	المهنا	../images/lawyer2.jpg	0579	accept	male	../images/AlShaimaaYahya.pdf	Lawyer5
mai.co	مها	محمد	../images/lawyer3.jpg	055	wait	female	../images/AlShaimaaYahya.pdf	Lawyer3
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 65: Database after deleting lawyer

- **Manager accept/decline lawyer:** this interface allows for manger to view lawyer's cv and accept/decline lawyers, as shown in Figures 66, 67, 68, 69,70 below.

lawyer_email	first_name	last_name	photo	phone	state	sex	cv	lawyer_id
Fahad@mail.com	فهد	المهنا	../images/lawyer2.jpg	0579	accept	male	../images/AlShaimaaYahya.pdf	Lawyer5
mai.co	مهند	محمد	../images/lawyer3.jpg	055	wait	female	../images/AlShaimaaYahya.pdf	Lawyer3
salwa.alhajjaji@...	سلوى	الحجاجي	../images/0oFL2EIh_...	0567...	wait	female	../images/AlShaimaaYahya.pdf	Lawyer6

Figure 66: database before accepting lawyer

Figure 67: view lawyer's request

نموذج السيرة الذاتية المعتمد

١ - المعلومات الشخصية :
الاسم /.....
رقم الهوية ( )
مكان الميلاد /..... وتاريخه / /
الحالة الاجتماعية /.....
الجنسية/.....
العنوان /.....
المدينة و المحافظة/.....
رقم الهاتف ( )
رقم الجوال ( )
البريد الالكتروني /.....

٢-التعليم او المؤهلات:
وتضم المؤهلات التي حصلت عليها وجهاتها وترتب الحديث أولاً ثم الأقدم فالأقدم

Figure 68: lawyer's CV

lawyer_email	first_name	last_name	photo	phone	state	sex	cv	lawyer_id
Fahad@mail.com	فهد	المهنا	../images/lawyer2.jpg	0579	accept	male	../images/AlShaimaaYahya.pdf	Lawyer5
mai.co	مهند	محمد	../images/lawyer3.jpg	055	wait	female	../images/AlShaimaaYahya.pdf	Lawyer3
salwa.alhajjaji@...	سلوى	الحجاجي	../images/0oFL2EIh_...	0567...	accept	female	../images/AlShaimaaYahya.pdf	Lawyer6

Figure 69: database after accepting lawyer

Figure 70: accepting lawyer email

Reference:

[1] Available at: https://law.freeadvice.com/ [04 October 2018].

[2] Available at: https://rossintelligence.com/ [04 October 2018].

[3] Available at: https://www.ailira.com/ [05 October 2018].

[4] Available at: https://www.laboreducation.gov.sa/Ar/Pages/UserConsultationLanding.aspx [21 October 2018].

[5] Available at: https://itunes.apple.com/app/id1427999657 [01 November 2018].

[6] Available at: http://robotlawyerlisa.com/ [01 November 2018].

[7] Mandl, Thomas. (January 2008). Artificial Intelligence for Information Retrieval. University of Hildesheim, Germany. 151-156.

[8] Lai, Jianchang. Mu, Yi. Guo, Fuchun. Jiang, Peng. Susilo, Willy. (2018). Privacy-enhance d attribute-base d private information retrieval. School of Computer Science and Technology, Nanjing Normal University, Nanjing, China. 275–291.

[9] Available at: https://amitgunjal.wordpress.com/2016/11/21/the-process-of-information-retrieval-from-scratch/ [1 April 2019].

[10] Ralph Weischedel, Jaime Carbonell, Barbara Grosz, Wendy Lehnert, Mitchell Marcus, Raymond Perrault, White Paper on Natural Language Processing, 13.

[11] Bird, Steven. Loper, Edward. (2004). NLTK: The Natural Language Toolkit, Barcelona: Association for Computational Linguistics Stroudsburg, PA, USA, 4.

[12] Bird, Steven. (2017). NLTK Documentation Release 3.2.5, 83.

[13] Available at: https://hrdf.org.sa/Content/files/labor_system.pdf [04 October 2018].

[14] Available at: https://mobirise.com [Accessed 10 Feb. 2019].

[15] Available at: https://www.mysql.com [Accessed 7 Feb 2019].

[16] Available at: https://www.anaconda.com/what-is-anaconda/ [14 Nov 2018].

[17] Available at: https://www.livechatinc.com [Accessed 10 Feb. 2019]

[18] Shawar, BA. and Atwell, E. (2002). A comparison between Alice and Elizabeth chatbot systems. University of Leeds, School of Computing research report, 19.

[19] Shawar, BA. and Atwell, E. (2003) Using dialogue corpora to train a chatbot, England, School of Computing, University of Leeds, Leeds LS2 9JT, 10.

[20] Available at: https://agileintro.wordpress.com/2008/01/04/waterfall-vs-agile-methodology/ [16 Nov 2018].

[21] Sommerville. (2011). Agile software development. Software Engineering 9th Edition. Boston: Addison-Wesley 790

[22] Available at: http://pavel.surmenok.com/2016/09/11/chatbot-architecture/ [22 November 2018].

[23] Available at: https://www.spyder-ide.org/ (Access 28/03/2019)

[24] Available at: https://jupyter.org/ (Access 28/03/2019)

[25] Available at: http://flask.pocoo.org/docs/1.0/quickstart/ (Access 1/04/2019)

[26] Available at: https://byet.host [Accessed 5 Apr. 2019].

[27] Available at: https://www.heroku.com/what (Access 6/04/2019)

[28] Available at: https://www.gfotu.org/Faq.aspx (access:02-03-2019)

[29] Shawar, Bayan. Atwell, Eric. (2007). Different measurements metrics to evaluate a chatbot system. 7