“A Mobile Speech Interface for Emergency Help: EI Siri”

Yanchao Yu
H00099161

August 2013

Computer Science
School of Mathematical and Computer Sciences

Dissertation submitted as part of the requirements for the award of the degree of MSc in Software Engineering
“A MOBILE SPEECH INTERFACE FOR EMERGENCY HELP: EI SIRI”

School of Mathematics and Computer Science
– MSc Dissertation –

Supervisor: Oliver Lemon

Yanchao Yu (H00099161)
DECLARATION

I, Yanchao Yu, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: ..................................................................

Date: ..............................................................................
I would like to thank all people who were supporting me and providing me lots of helpful suggestions in my project. Firstly, I want to rend many thanks to my supervisor, Professor Oliver Lemon, who guided me and helped me to deal with problems in this project. Working with him is a really great experience. He makes me to know how to start a research in a new field.

Secondly, I want to thank Dr Helen Frances Hastie, who is my second reader in my project. She analysed my project proposal and provided some worthy advices. The feedback from her helps me to improve my project. She also helped me to deal with the core problem of my project.

During this project, I learned a lot from them, not only in terms of professional skills (e.g. core technology and skills to cope with problems), but also in terms of rigorous research attitude, especially when it comes to scientific research methodology.
**ABSTRACT**

With the increasing attractiveness of Human-Computer Interaction (HCI) and mobile applications, people’s demands on advanced Spoken Dialog Systems (SDS) have increased rapidly. Modern dialog system on PCs or mobile devices could present a practical natural language conversation based on a natural language processing interface. The voice-based applications attempt to perform tasks and provide some responses according to users' audio commands or queries. They are trying to affect people's life, not only on aspect of research, but also on aspects of business, education, and even healthcare. Unfortunately, the majority of voice-based applications are based on question-answering mechanisms rather than a real conversational mechanism. Therefore, it is harder to provide a practical assistance for individuals, especially in terms of medical emergency situations.

This dissertation proposes a dialog system which focuses on developing a mobile voice interface based on Android platform. The purpose of this project is to provide users first aid in emergency situations. This application could be an intelligent emergency assistant helping users to diagnose patients’ states and providing corresponding help. The report introduces principal voice-based technologies, such as SDS components and VoiceXML techniques. This mobile application is developed for Android mobile devices by using the Eclipse IDE and Google APIs. At the end of project, I evaluated whether the proposed mobile voice interface is easier and faster to be used for emergency help, compared to other first aid related applications or websites.

**Key Words:** Human-Computer Interaction, Spoken Dialog System, natural language processing, question-answering mechanism, conversational mechanism, emergency, SDS components, VoiceXML, Google APIs.
# Table of Content

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>i</td>
</tr>
<tr>
<td>Acknowledgment</td>
<td>i</td>
</tr>
<tr>
<td>Abstract</td>
<td>i</td>
</tr>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Objectives</td>
<td>2</td>
</tr>
<tr>
<td>Literature review</td>
<td>4</td>
</tr>
<tr>
<td>3.1 Spoken Dialogue Systems (SDS)</td>
<td>4</td>
</tr>
<tr>
<td>3.1.1 Introduction to SDS</td>
<td>4</td>
</tr>
<tr>
<td>3.1.2 Spoken Dialogue System Classification</td>
<td>4</td>
</tr>
<tr>
<td>3.1.2.1 Functional Classification</td>
<td>5</td>
</tr>
<tr>
<td>3.1.2.2 Technical Classification</td>
<td>5</td>
</tr>
<tr>
<td>3.1.3 Architecture of SDS</td>
<td>8</td>
</tr>
<tr>
<td>3.1.3.1 Automatic Speech Recognition (ASR)</td>
<td>10</td>
</tr>
<tr>
<td>3.1.3.2 Natural Language Understanding (NLU)</td>
<td>12</td>
</tr>
<tr>
<td>3.1.3.3 Dialog Manager (DM)</td>
<td>12</td>
</tr>
<tr>
<td>3.1.3.4 Natural Language Generation (NLG)</td>
<td>15</td>
</tr>
<tr>
<td>3.1.3.5 Text To Speech (TTS)</td>
<td>17</td>
</tr>
<tr>
<td>3.1.4 Speech Processing Theory &amp; Technique</td>
<td>18</td>
</tr>
<tr>
<td>3.1.4.1 Natural Language Processing Theory</td>
<td>19</td>
</tr>
<tr>
<td>3.1.4.2 Voice XML</td>
<td>21</td>
</tr>
<tr>
<td>3.1.5 Challenges</td>
<td>22</td>
</tr>
<tr>
<td>3.2 Mobile Voice Applications</td>
<td>22</td>
</tr>
<tr>
<td>3.2.1 Siri on iPhone</td>
<td>23</td>
</tr>
<tr>
<td>3.2.2 Nina on Android</td>
<td>23</td>
</tr>
<tr>
<td>3.2.3 Google Now</td>
<td>24</td>
</tr>
<tr>
<td>3.2.4 Utter</td>
<td>25</td>
</tr>
<tr>
<td>3.3 Application &amp; Websites for first aid</td>
<td>25</td>
</tr>
<tr>
<td>3.3.1 First aid by British Red Cross</td>
<td>25</td>
</tr>
<tr>
<td>3.3.2 British Red Cross</td>
<td>26</td>
</tr>
<tr>
<td>3.3.3 NHS Choices</td>
<td>27</td>
</tr>
<tr>
<td>3.4 Android API Technique</td>
<td>28</td>
</tr>
<tr>
<td>3.4.1 Google Speech API</td>
<td>28</td>
</tr>
<tr>
<td>3.4.2 Google Maps &amp; Location APIs</td>
<td>29</td>
</tr>
<tr>
<td>Methodology</td>
<td>30</td>
</tr>
<tr>
<td>4.1 System Architecture</td>
<td>30</td>
</tr>
<tr>
<td>4.2 Development Methodology</td>
<td>31</td>
</tr>
</tbody>
</table>
9.5 GUI Implementation .......................................................................................... 72
  9.5.1 Splash ........................................................................................................ 72
  9.5.2 EI Main Activity .......................................................................................... 72
  9.5.3 EI Clinic ...................................................................................................... 73
  9.5.4 EI End ......................................................................................................... 76
System Testing and Evaluation ............................................................................. 78
  10.1 Overview ...................................................................................................... 78
  10.2 Testing Description ....................................................................................... 78
  10.3 Evaluation .................................................................................................... 79
    10.3.1 Participant Background ......................................................................... 79
    10.3.2 Result analysis ....................................................................................... 80
    10.3.3 Evaluation Discussion .......................................................................... 85
    10.3.4 Improvement Based On Evaluation ....................................................... 86
Summary .................................................................................................................. 88
  11.1 Overview ...................................................................................................... 88
  11.2 Achievement & Limitation ......................................................................... 88
    11.2.1 Achievements ....................................................................................... 88
    11.2.2 Limitations ........................................................................................... 88
  11.3 Future Work ................................................................................................ 89
  11.4 Conclusion .................................................................................................. 90
Bibliography ............................................................................................................ 91
Appendix A: Questionnaires for the “EI Siri” ....................................................... 94
Appendix B: Voice-Base XML Files in the “EI Siri” ............................................. 97
LIST OF TABLES

Table 1 Example about Error handling [5] ................................................................. 14
Table 2 User Requirements for “Emergency Intelligent Siri” .................................. 34
Table 3 System Requirements for “Emergency Intelligent Siri” ............................. 36
Table 4 Risks in Development of Mobile Voice Interface Project .......................... 44
Table 5 Text Description for Application ................................................................ 48
Table 6 Eight Golden Rules of Interface Design [37] .............................................. 57
Table 7 Testing Descriptions .................................................................................... 78
Table 8 Collection of Answers for Question A ....................................................... 81
Table 9 Collection of Answers for Question B ....................................................... 82
Table 10 Collection of Answers for Question C .................................................... 83
List of Figures

Figure 1 Example of Assistant part in Emergency (a) Slight (b) Dangerous .................. 3
Figure 2 Example of Diagnosis part in Emergency .................................................. 3
Figure 3 Functional Classification of SDS [6] ......................................................... 5
Figure 4 Technical Classification of SDS [7] ......................................................... 5
Figure 5 Example of simplex linear script [7] ......................................................... 6
Figure 6 Example of State Transition Network [7] ................................................ 7
Figure 7 Traditional Architecture of Spoken Dialog System [6] .............................. 8
Figure 8 Speech Recognition Architecture [8] ....................................................... 10
Figure 9 Example of CFG Grammar in NLU [12] .................................................. 12
Figure 10 Example about Roles of DM in Restaurant Dialog [5] ............................. 13
Figure 11 The Process of NLU and NLG [14] ....................................................... 15
Figure 12 The Reference Architecture for NLG System [12- 15] ............................ 16
Figure 13 Process of Text-To-Speech Conversion [17] .......................................... 17
Figure 14 Flow of Voice-Processing in DM [19] ................................................. 19
Figure 15 The Layers of Natural Language Processing [7] ...................................... 20
Figure 16 Architecture Model for VoiceXML [6- 22] ............................................. 21
Figure 17 First Aid by the British Red Cross[31] .................................................. 26
Figure 18 First Aid for Heart Attack in British-Red-Cross Website [2] ...................... 26
Figure 19 Information about emergency of “Burns and scalds” on NHS Choices [1] 27
Figure 20 The Components in Google Speech API package [32] ............................ 28
Figure 21 The Fundamental Architecture of Emergency Intelligent Siri ................ 30
Figure 22 Development Methodology: Iterative Incremental Development ........... 31
Figure 23 Information Form in Evaluation ............................................................ 32
Figure 24 Case Cards in Evaluation Phase ............................................................ 33
Figure 25 Example of Questionnaire for Evaluation Phase ................................. 33
Figure 26 Example of Final result in the Form of Text in Application ................... 35
Figure 27 Example of Voice Mode in Application ............................................... 36
Figure 28 Example of System-initiative Mechanism ............................................ 38
Figure 29 Gantt chart of Project Schedule ............................................................. 43
Figure 30 Basic Architecture of “EI Siri” ............................................................... 45
Figure 31 Activity Diagram of “Emergency Intelligent Siri” ................................. 46
Figure 32 Use Case Diagram on “Emergency Intelligent Siri” .............................. 47
Figure 33 Class Diagram of ”Emergency Intelligent Siri” .................................... 49
Figure 34 Class Diagram of Dialog Manager in “EI Siri” ..................................... 51
Figure 35 XML Files Structure in “Emergency Intelligent Siri” .......................... 52
Figure 36 Dialog Tree of General XML File ....................................................... 54
Figure 75 Analysis of Answers for Question A ......................................................... 81
Figure 76 Analysis of Answers for Question B .......................................................... 82
Figure 77 Analysis of Answers for Question C .......................................................... 83
Figure 78 Satisfaction of Information using Different Applications ....................... 84
Figure 79 Time Cost Using Different Applications ................................................... 85
Figure 80 Example of Enlargement of Vocabulary ................................................... 87
INTRODUCTION

Spoken Dialog systems (SDS) play increasingly important roles in affecting people’s life. Dialog systems could provide a set of natural and continuous interactions between machines and human based on a series of SDS components (e.g. Automatic Speech Recognition and Dialog Manager), and voice-related techniques (e.g. Natural Language Processing and VoiceXML). Through the latest theories and processing algorithms, current dialog systems have been significantly improved. The improvement of SDS promotes the innovation in the domain of artificial intelligence. In the past decade, an increasing number of voice-based applications are introduced by some IT companies on computers, especially on mobile devices. These voice applications, such as Siri, Google Now, Nina and Utter, have dramatically changed some current individuals’ lifestyle, not only on research, but also on business, education and so on.

However, the majority of voice-based applications cannot support a real dialog, possibly because of the lack of more advanced techniques on software and hardware. It means that they can accomplish some tasks users require based on a question-answering mechanism, but not keep continuous interactions with users. The problem restricts the improvement of voice-based software in the intelligent field, such as intelligent emergency assistant. Without the assistant, people must spend more time on confirming the emergency state and searching for information on websites (e.g. NHS Choices [1] and British Red Cross [2]) to provide aid. It might bring a negative influence on life-saving.

For this reason, the project aims at developing a mobile voice application based on a natural language interface for emergency aid, called “Emergency Intelligent (EI) Siri”. It mainly focuses on the diagnosis of patient’s state and first-aid guidance. The proposed application will determine whether it could contact the Emergency Centre or not by automatically analysing seriousness of problems. On the other hand, the voice-based application could distinguish the states of emergency via a set of simple queries. In order to implement these functions, I will concentrate on some particular emergencies, including burns and scalds, heart attack, asthma attack, sprains and strains, as well as food poisoning. All processes of diagnosis will be based on system-initiative mechanism, which means users just need to answer "yes" or "no" for diagnosis related questions provided by “EI Siri”. This voice application should be developed with Google APIs by JAVA language, and supported on Android platform.

This report will be separated into 5 parts. The first section will introduce the objectives of this project. The second part is called literature review, in which the main dialog related techniques (e.g. SDS, VoiceXML and Google APIs), modern mobile voice applications (e.g. Siri, Nina and Google Now) and emergency related websites (e.g. NHS Choices [1] and British
Red Cross [2]) and applications will be explained in detail. The third part will describe the architecture of proposed application, development and evaluation methodologies. It also analyse all requirements of proposed application. There will be several issues and a detailed project plan described in the 4th part. The 5th part, as the core part of report will describe the design, implementation and evaluation of system. Finally, I will discuss this project in terms of achievements, limitations and future work in summary.

**OBJECTIVES**

Current mobile voice applications, such as Siri, Nina and Google Now, are more popular and important in life than ever before. They could provide users help on different aspects. For example, users could use these applications to manage their bank accounts by a set of voice command. They could also use applications to search for various information (e.g. learning materials, weather, and life skills). However, most applications cannot keep a continuous interaction with a user, because they are based on question-answer mechanism rather than the real conversations. Many systems cannot create an information network to help users to solve some relatively complicated problems or situations, such as Emergency.

In the emergency situations, there are some complex logical structures between different states. People without an intelligent interaction need to build the relationships between conditions about different states, and then find out the results as well as provide corresponding solutions via a lot of relevant materials searched online.

Therefore, the main objective of this project is to propose a mobile voice-based application for emergency situation, called "Intelligent Emergency Assistant". It could assist individuals to deal with some potential medical emergencies. It will be a good partner for first responders. In this project, I will test the hypothesis that “Using proposed mobile spoken emergency application is faster and better to provide aid than using existing web-based first-aid websites”.

Actually, this objective described could be divided into two parts: Diagnosis part and Assistant part. Assistant part aims at providing guidance about first aid to teach users how to help themselves or others who are in the emergency situations especially when they are already aware of their situations. For example, once a user finds that someone is burned or scalded, he or she could ask for help from "emergency assistant". Their possible dialog is shown below (Figure 1).
“A Mobile Speech Interface for Emergency Help: El Siri”
Yanchao Yu (H00099161)

In the figure 1, system will analyse the level of problem via some simple questions firstly. If his or her level is slight, system will teach user how to deal with effected area in the form of voice. But if the level is dangerous, system will help users to contact Emergency Centre in parallel with providing guidance for medical aid.

Another sub-objective of application is Diagnosis. The purpose of this part is to confirm the situation of emergency via a set of queries before providing help. It will determine and provide the appropriate guidance to help users once system precisely diagnoses the situation of a man. For example, user need to ask helps from "emergency assistant" when he finds a man lying on the ground but doesn’t know how to help him. The possible dialog is introduced in the following (Figure 2).

As shown in the figure 2, system could confirm that the patient has an asthma attack by asking several simple questions, and then could provide relevant medical guidance to users.

FIGURE 1 EXAMPLE OF ASSISTANT PART IN EMERGENCY (A) SLIGHT (B) DANGEROUS

FIGURE 2 EXAMPLE OF DIAGNOSIS PART IN EMERGENCY
**LITERATURE REVIEW**

### 3.1 Spoken Dialogue Systems (SDS)

#### 3.1.1 Introduction to SDS

The past two decades could be seen as an essential phase for the technology of ‘Human-Computer Interaction’ (HCI). HCI involving study, planning as well as designing of interaction between people and computers was developed and improved rapidly [3] in the past 20 years. There are a significant number of applications with HCI created in different fields. These applications with HCI, so-called ‘spoken dialogue systems’, constantly affected peoples’ life in recent years. According to Glass [3], spoken dialogue systems (SDS) could be viewed as a computer application providing the communication interface between machine and users. They provide a bridge between users and a system using voice user interface (VUI). Pucher [4] also thought that among the most advanced form of VUI might be spoken dialogue systems, which allow speech interaction. The communication interface could take natural speech from human as input and generate relevant responses as output by analysing and managing dialogs. It also could perform a set of tasks required by people (users).

Due to the invention of intelligent dialog system, people start to redefine Human-Computer relationships. Computers are more likely to be an assistant or a partner in human’s life rather than a responder to people. Computers will take a more active position in different fields, not only in research but also in commercial life.

In terms of research, scientists attempt to develop and improve highly intelligent machines with robust SDS technique, so-called robot, which could communicate with human and perform various complex tasks people cannot complete.

On the other hand, an increasing number of spoken dialogue systems have been developed for commercial purpose since the early 1990s [5]. Voice applications such as information search systems and traffic navigation systems enable users to conveniently find out what they want in an extremely short time.

#### 3.1.2 Spoken Dialogue System Classification

Due to the improvement of spoken dialogue techniques, there are a significant number of SDSs classified in different ways. Among the main classifications of SDS are functional classification and technical classification.
3.1.2.1 Functional Classification

In the functional aspect, Jokinen [6] argued that the main two types of SDSs are task-oriented and non-task-oriented, as shown in Figure 3. She showed that task-oriented dialog systems could assist users to perform tasks they required, whether simple or complex. These systems take strong directness to accomplish specific tasks, such as flight-searching, hotel-booking, or managing a bank account, planning an individual holiday. On the other hand, non-task-oriented systems could be defined as a communication subject without any practical tasks in a conversational process. However, Jokinen [6] also indicated that both of systems, task-oriented and non-task-oriented, could not be completely independent of one another. It means that it is difficult to distinguish task-oriented dialogue systems with non-task-oriented ones. For example, there may be simple conversations in the task-oriented system, and a non-task-oriented system is also likely to involve into a solving-task.

![FIGURE 3 FUNCTIONAL CLASSIFICATION OF SDS [6]](image)

3.1.2.2 Technical Classification

In the technical classification, there could be three main types of spoken dialog systems, including pattern-response, state-based as well as plan-based dialogue systems [7], as shown in Figure 4.

![FIGURE 4 TECHNICAL CLASSIFICATION OF SDS [7]](image)
Pattern-response Systems

Pattern-response dialog systems follow the most fundamental rule in the SDS techniques, named pattern-response rules. Bickmore [7] showed that the applications with rules could match a sequence of natural words contained in user’s input utterances. It might generate a set of relevant output utterances once the match is found. According to Bickmore [7], this kind of dialog system has a series of skills (also named ‘tricks’) relating to the form of coherent dialog, rather than discourse context. Discourse context is built with strong pertinence according to different situations, such as educational events and healthcare. These tricks consist of system original dialogs, sense-making ability, as well as reflection of user’s inputs with little change. Bickmore [7] indicated that in the system original dialogs, a large number of possible outputs could help systems to answer open-ended questions from users. Sense-making ability might promote to unify the explanation of outputs. Because of these reflections without changes from systems, users are likely to have an illusion that system could understand what users said. Unfortunately, these tricks could not avoid dialogs from all problems [7]. It is difficult for systems to truly understand the meaning of what users talked about once users’ inputs are not constrained. Therefore, pattern-response dialog system could not cater for needs of more complex situations.

State-based Systems

State-based dialog systems are better at dealing with different situations than pattern-response ones. The state machine is a more common technology used in the state-based dialog system. This machine contains a set of states which could be reused or transferred from one to another one. In the early state machine, each state could represent a simplex linear script such as the one shown in Figure 5.

![FIGURE 5 EXAMPLE OF SIMPLEX LINEAR SCRIPT [7]](image)

In the state-based dialog system, state machines could create a large number of states and extend them to generate a complex network, so-called State Transition Network, in which each state may have more than one next state [7], as shown in Figure 6. Each dialog could include more sub-dialogs for the same question.
Because of the wide use of this kind of dialog system (state-based), there are more complex situations. In order to adapt the network to variations of situations, hierarchical state transition network, as a new dialog model, was introduced into state-based dialog system. It "serializes the state-segments in the state machine, and enables the system to accomplish multiple conversational functions" [7]. Bickmore [7] proved that the new model contains a stable database in which there are a large number of information relating to users’ utterances. State-based system constantly extends or improves the state model underlying another voice technique, VoiceXML, which will be explained in the speech processing theory section.

Plan-based Systems

Apart from pattern-response and state-based systems, plan-based systems are another popular dialog systems used widely. Obviously, users’ expression (utterance) could not always represent directly their real goals, plan-based dialog system seen these dialog segments as a part of task plan [7]. Systems could judge which task plan utterances belong to rather than just generate the responses to users. In order to improve the accuracy of judging users’ utterance, synthesizing the task plans and inferring user’s goals are necessary in complex conversations. Bickmore [7] showed that plan-based systems could generate dynamic planning and plan inference, which are extremely complex. He also pointed that although it could effectively deal with problems or accomplish any tasks contained in its task-planning, system needs to conduct a huge number of samples of simple dialogs and linked them with related task plans.
3.1.3 Architecture of SDS

In recent decades, there are a significant number of technologies developed for supporting the development of SDS. Scientists created different approaches and patterns to improve their efficiency. However, as Jokinen [6] said, spoken dialog systems will be designed and implemented by following the traditional architecture of SDS (as shown in Figure 7). It is viewed as a prototype of SDS supporting speech related activities.

![Figure 7: Traditional Architecture of Spoken Dialog System](image)

The figure above indicates that there are three main parts in the fundamental process of SDS, such as speech understanding, dialog managing as well as system output. They are described in the following.

**Speech Understanding**

Speech understanding consists of two processes:

1. Automatic Speech Recognition (ASR)
2. Natural Language Understanding (NLU)

Lee [5] pointed out that ASR plays an essential role in performing speech to text conversion. It could identify words from voice inputs and sends them to next component, NLU.

NLU, also called Spoken Language Understanding, focuses on analysing the textual input from ASR using a set of natural language processing techniques (e.g. morphology-analysing and part-of-speech tagging) [3-4-5]. It carries out a basic analysis activity, mapping utterances to semantic frame and extracting user goals and dialog actions form semantic frame.
Dialog Managing

Among the most essential one of all components is the dialog manager (DM). It may be seen as the core of SDS [4], involving controlling all activities relating to other components. The main purpose of this component is to link parts of speech understanding and system output. There are three steps in the dialog manager [6], as listed below:

1. Receiving processed inputs from ASR and NLU.
2. Interacting with more external information.
3. Producing textual output to end-users

Apart from these steps mentioned above, the dialog manager (DM) also focuses on controlling the whole flow of SDS.

According to Jokinen [6], DM consists of two main components, a dialogue context model and dialogue control. Dialogue context model is responsible for following the tracks of relevant information in dialog to support the process of dialog management. It could record all information about what is said and share their extensions between users and dialog systems. On the other hand, Dialogue control makes the decisions about what and how to do next in the dialog process [6]. It is related to both of the users’ inputs and outputs to them. The dialog manager (DM) will be described in detailed in the following section of dialog manager.

System Output

In the final part, System output consists of two components, Natural Language Generation (NLG) and Text to Speech (TTS).

Natural Language Generation (NLG) aims at generating the corresponding textual answers in natural language underlying the knowledge database [5]. It needs language skills and knowledge about different types of words to sequence the relevant information before sending response to final component.

Text-to-Speech could be seen as the back-up process of ASR. It might synthesize the utterance and generate the system output in voice [5]. Users could receive the corresponding speech answer from hardware device.

Significantly, there could also be some variations on the method of processing data in each part and the forms or types of data transferred between components. The following sections will introduce and explain each component.
3.1.3.1 Automatic Speech Recognition (ASR)

As mentioned in the last section, automatic speech recognition (ASR) is one of the key components in development of spoken dialog system [5]. The aim of ASR is at identifying the audio words from users. In other words, it could promote the conversion of acoustic signals received by input devices, such as microphone or mobile phone [8]. ASR could provide the basic mathematical formula to get the more reasonable word sequence, as shown below:

\[
\hat{W} = \arg \max_{W \in \mathcal{L}} P(O|W)P(W)
\]  
(Formula a)

In this Formula (a) introduced by Karpagavalli [8], "O" represents the acoustic observation and "W" represents corresponding word sequence. The goal of this formula is to find the maximum posterior possibility \( P(W|O) \) of each word in the sequence. The \( P(W|O) \) could be represented by the prior possibility \( P(W) \) and the likelihood of audio observation \( P(O|W) \).

In a review of automatic speech recognition [8], it describes several types of ASR, such as isolated words, connected words, continuous speech as well as spontaneous speech. Isolated word could be an independent word or utterance, which could be sampled on both sides [9]. To be similar with isolated words, connected words could be processed as a single sample, but also as a string of utterance with tiny intervals between each two of them [8]. Continuous speech could receive coherent and complete speech in specific limits while spontaneous speech sounds more like a natural speech without any preparations [10].

Speech recognition technique consistently follows the basic architecture model (as shown in Figure 8), though it is improved rapidly. In this model, there are several main steps (e.g. sampling, acoustic front-end, and decoder) [8].

![FIGURE 8 SPEECH RECOGNITION ARCHITECTURE [8]](image_url)
In the first step, it focuses more on obtaining at least two samples of spoken utterance with similar frequency in each cycle. ASR could use these samples to measure positive and negative parts of voice-wave individually [8]. Hence, the more samples it received, the more accurate results ASR could get.

The architecture shows that the acoustic font-end step consists of three phases, pre-emphasis, windowing and feature executing. Pre-emphasis could provide more information to next step and improve its performance by increasing energy in high frequencies of audio segments [8]. The part of windowing recognizes a set of processing signals (called windows) and their data(named frames) to promote the extraction of features [8]. In this extraction phase, there will be a number of methods to get feature vectors, (e.g. principle component analysis method and linear discriminate analysis method).

The final step is called the decoder, in which it accomplishes the task of speech recognition with acoustic model and a language model. The acoustic model aims at building a statistical model for speech utterances to estimate its observation likelihood \( P(O|W) \) mentioned above [8]. A language model describes a set of linguistic limitation to forecast the words’ possibility in a sequence [8].

On the other hand, the review of ASR [8] provides a significant number of approaches used to enhance ASR process. Among one of the most essential and widely used approaches is Hidden Markov Model (HMM) developed in the early 1980s [8]. HMM contains three basic problems, evaluation problem for computing possibility of observation sequence, hidden state determination for optimizing relevant state sequence, as well as learning problems for optimizing the model parameters [11]. ASR also provides related algorithms for solving these problems. It uses forward algorithm for evaluation problem, Viterbi algorithm for decoding one and BaumWelch algorithm of learning [11].

Karpagavalli [8] pointed out that ASR applies a Word Error Rate (WER) to measure its performance by recognizing the different length of word sequence via DYA (Dynamic String Alignment). It is a part of Levenstein distance working at word level rather than phoneme level. WER could be calculated via formula (b).

\[
\text{Word error rate (\%)} = \frac{100 \times \text{Insertion(D)+Substitution(S)+Deletion(D)}}{\text{No of reference words (N)}}
\]  
(Formula b)

Finally, system might use WRR (Word Recognition Rate) to report the performance of ASR instead of WER, such as formula c [8].

\[
\text{WRR} = 1 - \text{WER} = \frac{N - S - D - I}{N}
\]  
(Formula c)

[11]
3.1.3.2 Natural Language Understanding (NLU)

A Spoken Language Understanding module consists of two parts, an ASR component and a NLU component. NLU is viewed as one of main components in spoken dialog systems referring to understand the meaning of input utterance from ASR and generate the suitable semantic representations for conversational tasks [12]. In order to produce these representations, dialog systems are based on two popular grammars, general-purpose unification grammars and semantic grammars [12]. Comparatively, current dialog systems focus more on semantic grammars than general-purpose unification grammars.

Semantic grammars are defined as a kind of semantic analysis tool building the correspondences between semantic entities [12], such as CFG. CFG (Context-free Grammar) is defined as a formal grammar in formal language theory, in which each production rule will be of the form $V \rightarrow w$. In this form, "V" represents a "single nonterminal symbol", and "w" is a string of terminals.

In CFG, a set of rules in the left-hand side have corresponding semantic entities, as shown in Figure 9. A grammar converts the input utterance into the form of context-free grammar and separates them into different units according to their correspondences [12].

![FIGURE 9 EXAMPLE OF CFG GRAMMAR IN NLU [12]](image)

Apart from CFG grammar, semantic grammars also could decompose the sentences using semantic HMM grammar (described in the ASR section) [12].

3.1.3.3 Dialog Manager (DM)

In modern spoken dialog systems, DM is always defined as the heart of SDS architecture. It could process users true intention related to the input from NLU and response output for NLG. DM plays a set of essential roles in the dialog process [5], as shown in Figure 10 and described below:

[12]
“1. Finding and presenting possible results for users’ intention by linking to external database, in which there is mass knowledge of user input and discourse context.  
2. Getting more detailed information and confirming unclear information by sending appropriate questions.  
3. Forecasting the next action of system at a conceptual level to generate and send out system utterances in NLG and TTS.  
4. Managing generic conversational mechanisms (e.g. multi-layer dialogue) to improve a spoken system.”

According to Lee [5], a dialog could be composed of a series of circulations between users and SDS, which is a special process of exchanging information. The success of exchanging information in turns depends on different degrees of dialog initiative, including system-initiative, user-initiative as well as mixed initiative. In the system-initiative dialogs, system could accept more information from users by asking more than one question. Since system owns a number of limitations in ASR and NLU, the inputs of users could be determined in advance. In user-initiative one, users might determine what questions they want to ask and systems need to answer them. Although system may also provide some questions to confirm the unclear slots, users control the whole dialog as a leader [5]. To compare with others above, the mixed-initiative degree is a more flexible degree, in which system could manage conversation while users may provide more information or change requirements at any time. In order to receive further and clearer slots information, DM may handle more flexible conversational flows to cope with different situations [5].

Meanwhile, there are still some problems or errors from ASR or NLU in a spoken dialog system. DM might deal with a set of errors about inputs via error handling approaches for different strategies, including explicit confirmation, implicit confirmation and rephrasal [5]. There is an example for them listed in Table 1.
<table>
<thead>
<tr>
<th>Strategy Types</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explicit</strong></td>
<td><strong>Confirmation</strong></td>
</tr>
<tr>
<td>User: I want to go to Korean restaurant in New York.</td>
<td>System: So you want to go to Korean restaurant in New York?</td>
</tr>
<tr>
<td></td>
<td>Please answer yes or no.</td>
</tr>
<tr>
<td></td>
<td>User: Yes.</td>
</tr>
<tr>
<td><strong>Implicit</strong></td>
<td><strong>Confirmation</strong></td>
</tr>
<tr>
<td>User: I want to go to Korean restaurant in New York.</td>
<td>System: What is the name of Korean restaurant in New York?</td>
</tr>
<tr>
<td></td>
<td>User: Mapo BBQ.</td>
</tr>
<tr>
<td><strong>Static</strong></td>
<td><strong>Rephrasal</strong></td>
</tr>
<tr>
<td></td>
<td>User: I'd like to go to Korean restaurant in New York.</td>
</tr>
<tr>
<td><strong>Dynamic</strong></td>
<td><strong>Rephrasal</strong></td>
</tr>
<tr>
<td>User: I want to go to Korean restaurant in New York.</td>
<td>System: Sorry. Would you please rephrase using expression “I'd like to</td>
</tr>
<tr>
<td></td>
<td>go to [RES-TYPE] in [LOC-ADDRESS]”.</td>
</tr>
<tr>
<td></td>
<td>User: I'd like to go to Korean restaurant in New York.</td>
</tr>
</tbody>
</table>

**TABLE 1 EXAMPLE ABOUT ERROR HANDLING [5]**

In the explicit confirmation approach, a system could provide some specific questions to users for confirming some particular goals (e.g. name, time and location) [5]. Normally, users are only allowed to answer them with “yes” or “no”. While, in an implicit confirmation, system could repeat the part of input it understood in the next question to get further information. Lee [5] pointed out that both of explicit and implicit confirmation might be better at repairing the unreliable questions.

On the other hand, rephrasal consists of two types, static and dynamic rephrasals. The rationale of rephrasal is to accept a new utterance being similar with previous one to solve errors [5].

In order to enhance the DM, it is separated into three principal approaches to solve DM problems. (i) Knowledge-based DM might solve more problems via finite-state automata based on handcrafted rules [5]. Knowledge-based DM is widely used in many realistic spoken applications, since it is one the simplest approaches. (ii) Data-driven DM is more effective than knowledge-based DM, because it requires less time and effort [5]. Lee [5] indicated that data-driven approach could work automatically without human monitoring, though it needs time-consuming data annotation. This approach benefits the ASR and SLU components and promotes the implementation of standard dialog systems by reinforcement learning (RL) technique [13]. RL is viewed as a kind of machine learning technique concerning the dynamic changes in dialog modelling by optimizing rewards or cost features. (iii) Hybrid DM [5] is one of the more advanced approaches integrating the
techniques of reinforcement learning and supervised learning. The main aim of this approach is at optimizing dialog policies in traditional DMS.

### 3.1.3.4 Natural Language Generation (NLG)

Natural Language Generation (NLG) is defined as an essential module generating outputs in natural language from non-linguistic inputs [14]. NLG seems to be the inverse of NLU module, because NLU convert natural language inputs into meaning, while NLG could convert the meaning back into texts (shown in Figure 11) [14].

![Diagram of NLU and NLG](image)

**FIGURE 11 THE PROCESS OF NLU AND NLG [14]**

To be compared with NLU, NLG concerns more on the choices on different aspects, such as content selection, lexical selection, sentence structure and discourse structure [12-15]. For content selection, NLG process needs to choose suitable content from possible inputs based on a particularly communicative purpose [12-15]. NLG could find out the most suitable items for special notions based on lexical selection [12-15]. In the sentence and discourse structures, NLG might allot chosen contents into phrases or short sentences, decide how to involve discussed objects, and deal with multi-sentence discourse respectively [12-15].

Because the forms and contents of inputs may be different in NLG system, there is less likely to be a unified architecture for NLG [14]. However, the majority of practical systems frequently consider a general NLG architecture (Figure 12). This reference architecture has two principal components, discourse planner and surface realizer [12-15].
In the figure 12, among the most crucial components in NLG are a discourse planner and a surface realizer. A discourse planner is based on communicative goals and knowledge bases [12-15]. It could choose the appropriate contents from knowledge base and generate final discourse plan for communication in dialog by structuring these contents [12-15]. A discourse planner introduces two influential mechanisms, text schemata and rhetorical relations. Text schemata are built to define or specify the structures of text [12-15]. It could generate a high-level discourse structure for text expressing the sequence of actions or transitions. However, produced discourse structure contains a simple sequence of requests for generating sentences [12-15]. There are no advanced structures connecting different sentence together. Rhetorical relations, as one of the most important mechanisms, could cope with these problems based on RST (Rhetorical Structure Theory) [12-15]. RST is related to organizing text according to relationships between potions of text (e.g. sentences). NLG system might show all relations between these parts and reorganize output texts via rhetorical relations [12-15].

A surface realizer might accept all plans produced in previous components and produce single sentences following a set of grammars [12-15]. It could specify the possible outputs via these discourse plans. In the section of surface realizer, there are two most popular and important methods supporting its tasks [12-15]. They are systemic grammar and functional unification grammar. Systemic grammar, as a portion of Systemic-Functional Linguistics, views sentences as function set and keeps some rules for reflecting relevant functions [12-15]. It will map these functions via a set of realization statements from specified characteristics in systemic grammar. A functional unification grammar could perform and infer the functional structures using unification [12-15]. It is to produce generation grammar (so-called feature structure) with a number of possible alternatives and to integrate grammar with input standard by those feature structures. Both of systemic
grammar and functional unification grammar support the multiple levels in the process of generation [12-15].

3.1.3.5 Text To Speech (TTS)

Text-to-Speech (TTS) is seen as an essential phase in the spoken dialogue systems. The main objective of this phase is to generate voice speech by extracting and processing information from input text [16]. Currently, TTS systems are developed based on different principles, which could enhance the quality of speech, such as concatenative-speech synthesis [17] and corpus-based speech synthesis. Apart from them, rhythm also plays an important role in the implementation of TTS, because it could enable speech to be more natural and understandable [16].

However, there are still some problems in the TTS process. Among one of the most crucial problems is punctuation ambiguity [16], which is frequently occurred in the early TTS systems. In order to eliminate all errors, some researchers try to develop new TTS systems using a neural network [18]. But El-Imame [17] proposed a new process of conversion of text-to-speech, as shown in Figure 13.

In the figure 13, there are several phases in this new conversion, including text pre-processing, grapheme-to-phoneme transcription as well as synthesis-units generation.

Text pre-processing is an analysing part in text-to-speech process. It could analyse the input texts and provide the results to next phase, called grapheme-to-phoneme transcription [17]. The input text may consist of a set of special words (e.g. numbers, abbreviations, as well as acronyms), which will contribute to punctuation ambiguity speech [16-17]. In order to cope with it, this phase might process this information with exception lexicon comment, number rules. Exception lexicon is responsible for transiting input text to pronounceable forms and
separating them into several segments by comma, space or new-line [17]. The number rules could convert numbers into appropriate words in the real order [17].

The second phase is grapheme-to-phoneme transcription or phonetic analysis, in which the orthographical symbols could be converted into the phonological ones in different ways [17]. One of the most popular approaches is dictionary based approach, which could keep all kinds of words with right pronunciations [16]. It enhances the quality of pronunciation because of its speediness and high accuracy. Rule-based approach is another popular way to improve the pronunciations [16]. Word could generate its pronunciation based on some rules, such as letter-to-sound rules or a N-Gram Language model [17]. It is more complex than dictionary-based rules.

In the phase of synthesis units generation, the generated speech will be spoken in prosody, which consists of stress, rhythm and intonation [16]. Prosody is related to emotions of speakers. Modelling different intonations and prosodies could enhance speech intelligibility and naturalness [16]. Otherwise, acoustic processing in the phase could enable speech to be more humanized. There are several types of acoustic synthesis used in current TTS system (e.g. concatenative, formant and articulatory synthesis) [16]. Among the most popular one is concatenative synthesis. It requires a large database with all pre-recorded words. Concatenative synthesis could provide as natural sounding speech as human speaks by processing these words in database [16].

Finally, some researchers think that the next generation of TTS system will focus more on the emotional speaking styles [16]. It will make machines (e.g. computers and robots) able to express as rich sentiments as human has.

3.1.4 Speech Processing Theory & Technique

According to Glass [3], DM (Dialog Manager) could manage or control whole process in SDS. DM is viewed as the bridge linking the inputting and outputting part. Glass [3] showed that there are several main techniques supporting the implementation and improvement of spoken dialog system. Researchers attempt to cope with potential problems they will encounter in the development of SDS. The dialog system follows a typical flow of processing audio information in dialog manager [19], as shown in Figure 14.
In the flow chart of DM, it shows that among the most core technology or theories in the DM flow are Natural Language Processing (NLP) and VoiceXML, as mentioned in previous sections. Bickmore [7] thought that the main purpose of NLP is to divide the problems in the understanding and generation of utterance in natural language into several layers, as well as solve the sub-problems via advanced algorithms. The details about NLP will be explained in the next section.

On the other hand, VoiceXML is also one of the most widely used scripting languages for supporting IVR (Interactive Voice Response) [20]. It could be seen as a special form of HTML with voice nodes [21]. The difference between VXML and HTML is that HTML will suppose the graphic browser with input by keyboard and mouse and output by displaying on screen, while VXML supposes the voice browser with voice inputs from ASR and outputs for TTS mentioned above [21]. VoiceXML (version 1.0) was launched by VoiceXML Forum [22], which is an international industry organization spreading the use of VoiceXML and its relevant technologies. There are some applications using VoiceXML technique, such as VoiceGenie and Nuance [23]. The following section will introduce the special language in detail.

3.1.4.1 Natural Language Processing Theory

Natural Language Processing (NLP) is one of the most essential approaches in the development of spoken dialog systems. It involves all components in the architecture of SDS, especially ASR and NLU. NLP and spoken dialog system share the same goal of analysing utterance from users as well as generating corresponding answers or commands speech
In order to improve the accuracy of responses, NLP normally decomposes potential problems of natural language understanding and generating into different layers with knowledge of language (as shown in Figure 15) [7]. During the development of SDS, the majority of applications could require different knowledge of language to process the voice-based information in different layers and components [12].

![The Layers of Natural Language Processing](image)

**FIGURE 15 THE LAYERS OF NATURAL LANGUAGE PROCESSING [7]**

According to figure above, NLP approach uses the knowledge of phonology to analyse a sequence of sounds in the ASR [12]. It could enable systems to understand these sounds at acoustical level [12]. Because of some variations in words, morphological analysis could separate phonemes generated in former layer into several units to get their meanings with knowledge of morphology [7-12]. For individual words, SDS may use a set of structural knowledge of language (e.g. syntax knowledge) to organize a number of words in appropriate order [12]. To accurately answer queries, systems might be also required to capture the relationships between words using the knowledge of lexical semantics (for the meaning of words) and compositional semantics (for geographical differences of words) [12]. Finally, pragmatic and discourse knowledge are used to generate the responds of questions at the end of processing [12]. They will perform a series of activities showing communicative intentions between machines and speakers [7].

Otherwise, traditional NLP is not suitable for all spoken dialog systems, because there are still some problems in the processing of natural language, such as ambiguity. Ambiguity is one of the most important problems in NLP. In order to resolve ambiguities (described in TTS section), part-of-speech tagging technique and word sense disambiguation are proposed and used in system [12].
3.1.4.2 Voice XML

VoiceXML, also called VXML, is a standard mark-up language launched by AT&T Bell Lab [23]. The mark-up language could define the voice mark-up for voice-based applications or devices. In order to understanding the rationale of VoiceXML, McGlashan and his colleagues [22] designed a basic architecture model for it. This model (as shown in Figure 16) could precisely explain what the VoiceXML is and how it works.

![VoiceXML Architecture Model](image)

**FIGURE 16 ARCHITECTURE MODEL FOR VOICEXML [6-22]**

This figure shows a standard three-tier architectural model for VoiceXML, in which there are three main components, including Document Server, VoiceXML Interpreter Context and Implementation Platform. In the VoiceXML application, the interactions between three parts could promote the operations of voice processing. Document server may be responsible for processing various requests from VoiceXML interpreter contained in VoiceXML interpreter context as well as generating corresponding VoiceXML documents replied to it.

VoiceXML interpreter context, as the core of architecture model, focuses more on monitoring the audio inputs (such as commands and queries) from users in parallel with VoiceXML interpreter [22]. It could detect and analyse the relevant information in received voice-data and provide audio responds to users VoiceXML [22-23]. Meanwhile, McGlashan [22] indicates that both VoiceXML interpreter context and VoiceXML interpreter are also responsible for managing actions in the final layer, implementation platform.

In implementation platform, it seems to be essential to produce a set of responding events and system events for users’ requests and system status or activities respectively, such as spoken for users and timer alteration for system.

[21]
In VoiceXML, application could have a set of interrelated documents, named VoiceXML Documents [23]. This documents consisting of some dialogs could be viewed as a tool forming a dialogic finite state machine [22]. Dialog, which will be introduced in detail in the next part, is defined as the units or elements, so-called top-level elements in the VoiceXML application. McGlashan [22] thought that no user could be independent outside a conversational state or a dialog at a time. However, he [22] also pointed that each document could perform as an independent application, while each of them may be capable to determine which next document it transit to. It means that the execution of document could not be stopped until there is not one successor dialog found [24]. Among the most important features in VoiceXML is transition between different documents or sub-dialogs. This transition in a multi-documents application might bring significant influence on the performance of VoiceXML [22].

3.1.5 CHALLENGES

Obviously, developing spoken dialog systems has a set of challenges, which could affect the capabilities of system. Data collection is one of the main challenges in the development of SDS [3]. Due to the face that the number of questions, the number of participants and the scale of issues are different in different dialog system, it requires a mass of ‘real’ data for users [3]. The larger the quantity of information for users is, the stronger the practicality of the dialog systems is. The second issue what developers often face is evaluation process in the development of SDS [3]. This process could prove whether proposed system is usable or not. However, it seems to be difficult to assess the performance of different systems, because traditional statistical methods could not reflect user satisfaction [3]. In the past decades, developers attempt to assess their systems by using different metrics (e.g. component evaluation and integrity evaluation). Components evaluation is defined as a kind of evaluation approach assessing the performance of each component in SDS, such as ASR, NLU and NLG [3]. Integrity evaluation could assess the overall performance of the spoken dialog systems. Finally, portability is also one of main challenge in developing SDS [3]. Current SDS may allow developers to make flexible ports for new languages or domains, in order to create more mixed-initiative and practical spoken dialogue systems [3]. To cater for the requirements, researchers try to modularize each component in SDS and make DM to be more independent [3].

3.2 MOBILE VOICE APPLICATIONS

In the past decade, the technology of spoken dialog systems is widely used in various fields, such as education, e-commerce and research. There are an increasing number of voice-based applications developed and improved on PC or mobile devices. With the widely
use of mobile phones, these applications on mobile devices such as Siri, Google Now, Nina and Utter bring more positive influence on people’s life. The following sections will introduce them in detail.

### 3.2.1 Siri on iPhone

Siri is one of earlier speech applications or knowledge navigators developed with a voice interface by SRI International in 2007 and only supported on Apple’s IOS [25]. This voice-based application works on all mobile devices of Apple Company in recent years. It constantly focuses on controlling or managing all actions on mobile via voice inputs. By the year of 2011, Apple Company integrated new version of Siri into Apple IOS, in which there are an increasing number of functions added into voice application [25], as listed in the following:

1. Calendar & Reminders
2. Weather & Clocks
3. Messaging & Emailing
4. Entertainment with music, movies,
5. Mapping & Web browsers

Although Apple Company attempts to improve Siri application, there are still some limitations, such as language supporting, voice recognition and geographical location [25]. For example, Siri could support some of languages (e.g. English, French and Italian) but not all. And it cannot recognize all voice inputs from users once his or her accent is strange, such as Scottish and other non-native speakers. On the other hand, the function of Siri is limited by geographical locations, which means that users could not find out any information about The United States in the UK.

### 3.2.2 Nina on Android

Nina is a Siri-like voice application developed by Nuance Communications Company [26]. Nuance Company [27] shows that it seems to be a comprehensive intelligent personal assistant combining various technologies, including Nuance speech recognition, voice biometrics, TTS and NLU technology. This new individual assistant is multi-platform software on mobile devices. It means that it could work on both of Android and Apple IOS [26]. To compare with Siri on iPhone, the most significant difference is that Nina converts the interactions between human and system into a natural conversational form rather than a question-answer form [26]. It creates a relevant environment for one dialog, in which it knows users and users also know it.
Otherwise, Nina could set users’ voice as a kind of biometric password by biometric recognition and analysis technologies [26]. In the application, users could sign in or out their personal account and accomplish various tasks only using voice inputs.

Similarly, Nina could perform as much functions as Siri do. However, Nina is widely used in more fields than Siri or other voice-based applications (e.g. finance business, education, and even healthcare), due to the fact that it is based on the cloud platform rather than the exclusive interface [26].

In the terms of medical assistant, Nina could provide a set of effective solutions to improve the accuracy of clinical results, to promote the extraction of useful data from clinical documents as well as train the medical teams to enhance the quality of documentation [27].

3.2.3 Google Now

Google Now is invented by Google Company in the early 2012. It is viewed as a new intelligent individual assistant application with voice interface on the next generation of Android [28]. It extends the original Google Search function on Android and promotes a series of actions responding the voice-based requests from users. The main function of Google Now is to accept queries or commands from users and to provide corresponding answers combing graphic interface and voice interface [28]. As a part of Google search function, it could search location, recommendation, write email or SMS in Voice as well as other searching queries. According to Google Company [28], as a robust voice assistant, Google Now provides a set of cards to display its relevant information in detail. These specific cards are listed below [28]:

1. Activity Summary
2. Traffic
3. Gmail
4. Weather
5. Events: new Appointment, Birthday
6. Travel information: Flights, Hotels, Restaurant booking

However, Google Now just focuses on the personal works, such as managing people’s day and searching relevant information online. There is no any direct contribution to medical situation from Google Now, although users could use it to find out some good ways to deal with emergency from other websites (e.g. NHS Choices [1] and British Red Cross [2]).
3.2.4 Utter

Utter is the latest virtual individual assistant developed by Brandall [29]. It is an offline voice-based application. It could normally work without a data connection [29]. Utter could allow users to control their devices with a series of better algorithms. There might not be any graphic UI installed in the mobile device. According to description from Google Company [29], users could activate utter application in different ways (e.g. shaking, waving or button-clicking). Utter provides fast offline voice-based functions via recognition technique to manage all features, such as location service, information service and searching service.

3.3 Application & Websites for First Aid

In the medical emergency, providing effective aid timely seems to be so important to save others’ life. In the document for first responder application, Howard [30] said:

“With a cardiac arrest, you only get about 10 minutes to help. While on average, it takes 7 minutes for first responders from a 911 call to arrive.”

Hence, in the past decade, there are an increasing number of applications (e.g. first aid app) and websites (e.g. NHS Choices [1] and British Red Cross [2]) supporting first aid, which are extremely necessary especially in the emergency situations. They will be introduced in detail in the following sections.

3.3.1 First Aid by British Red Cross

“First aid” is an emergency assistant application developed by the British Red Cross [31], as shown in Figure 17. It works on multi-platform, such as apple and android platforms. There are some functions in this application, such as preparing, emergency, learning, testing and information [31]. Emergency function is more likely to provide some simple life-saving advises to first responders with videos. In an emergency situation, it seems to be easier and faster for users to find out corresponding information online [31].

Otherwise, "first aid" is also an excellent education application, because of its learning function [31]. It could teach people what and how to save their others’ life step by step in different emergency. Users also use information function to search for new information about emergency situations and share his or her experience with others. Finally in the testing function, application could check how many things you know in learning part [31].
3.3.2 British Red Cross

The website of British-Red-Cross is built by a humanitarian aid organization in the UK. This organization is responsible for helping people in crisis and providing responses to various emergencies [2]. It has several trains for people for first aid. This website could provide a set of first aid related information, not only the news about emergency all over the world, but also some knowledge about how to be as a first responder to help others in emergency [2], such as heart attack (Figure 18).

![FIGURE 18 FIRST AID FOR HEART ATTACK IN BRITISH-RED-CROSS WEBSITE [2]](image)
This web page about the emergency of heart attack in the figure 18 shows more details about how to provide first aid, including videos and rescue instructions in form of text. People could help someone in emergency according to this information. Apart from heart attack, British-Red-Cross also lists the first aid for other emergency, such as broken bone, asthma attack as well as sprains and strains.

### 3.3.3 NHS Choices

“NHS Choices” [1] is a meaningful website developed by NHS. As the biggest health website, it aims at providing information for conditions, treatment, and health services. “NHS Choices” collects a significant number of medical articles providing professional materials and assistant [1]. They enable people to choose healthier lifestyle. On the other hand, the website of “NHS choices” provides huge amounts of data about emergency rescue, such as the features of particular emergency, corresponding first aid. People could confirm the situation of emergency and provide relevant aid according to this knowledge, such as the first aid in Figure 19 [1]. It also helps people to decide when they have to ask for medical assistance.

![NHS Choices](image)

**FIGURE 19 INFORMATION ABOUT EMERGENCY OF “BURNS AND SCALDS” ON NHS CHOICES**

[1]

[27]
The figure 19 shows a set of information about the features of “Burns and scalds” and when to get medical attention. “NHS Choice” is significantly useful for patients especially when people have death threats.

3.4 Android API Technique

3.4.1 Google Speech API

Google Voice API (Application Programming Interface), so-called the Google Speech API, is introduced by Google Company. It consists of two main parts, one for speech input (speech API) [32], and another one for speech output (text-to-speech API). Both of them could assist developers to build different SDSs on computer or mobile devices.

Google Speech API, as an essential interface for input, is responsible for providing the speech dialog related functions, such as receiving audio inputs, recognizing the speech and responding to users [32]. It consists of an interface and several classes supporting these functions, as shown in Figure 20. Among the most important component is "SpeechRecognizer", which provides all relevant services for speech recognition [32]. Other interface and classes could be viewed as the assistants accessing to the class of "SpeechRecognizer" to implement various tasks in recognition service. Otherwise, "Callback" class is used to accept and present all responses to users [32].

![FIGURE 20 THE COMPONENTS IN GOOGLE SPEECH API PACKAGE [32]](image)

The text-to-speech API as an interface for system output focuses on returning a speech data or file from a text [33]. The API consisting of 3 interfaces and 6 classes could control the whole process of TTS mechanism mentioned in the Test-To-Speech section [33]. The core of TTS API is "TextToSpeech" class, which will create a sound file for particular text processed [28]
and synthesized in class. The class of "UtteranceProgressListener" is defined as the fundamental class providing all events related to utterance-processing in TTS through the synthesis queue [33].

Finally, during the development of spoken dialog systems, they will make up an integrated recognition mechanism.

### 3.4.2 Google Maps & Location APIs

Google Company also provides some APIs supporting map application and GPS technique, Google Maps API and Google Location API. Maps API is defined as an application interface of Google Map Service proving all features based on map data [34]. It could perform all functions related to Google Map Server, such as displaying map, downloading data and touching gestures [34]. The core class in Google Maps API is "MapView" used to display and manage a map with relevant data from Google Map Service [34]. Google Location API support developers to access to Location Service providing data about users’ locations or routes between different locations. "LocationManager", as the core component, plays an important role in the Location API [34]. It allows users to process all information from Location Service and to capture user’s location.
**METHODOLOGY**

4.1 **SYSTEM ARCHITECTURE**

In order to implement a mobile voice-based application for emergency, this project creates a fundamental architecture to display the main components of this application, as shown in Figure 21.

![Figure 21](image)

**FIGURE 21 THE FUNDAMENTAL ARCHITECTURE OF EMERGENCY INTELLIGENT SIRI**

According to the architecture, this application, so-called “Emergency Intelligent (EI) Siri”, is supported on Android platform with Google APIs. Google APIs will provide all interfaces for various functions, such as speech API for recognizing speech, text-to-speech API for sending voice responding and location API for confirming users’ locations. This mobile application is developed with stand-alone version. Application is separated into two parts, one is an interaction with users, and another one is dialog manager. In the interaction part, it will focus on two mainly stages: Speech input and Speech output. User (so-called first responders) could provide voice input and receive voice responds in this part. The stage of speech input consisting of ASR and NLU components will receive users’ voice requests (such as emergency requests and “yes” or “no” answers) and send them to the part of dialog manager. Speech output combining NLU and TTS is used to present emergency related information, including diagnosis questions and first aid.

On the other hand, dialog manager could receive and analyse text input and respond to users. It is the core of this proposed application and focuses on analysing all inputs from users to confirm emergency situations, diagnose illness levels and search for relevant rescue...
measures with VoiceXML files. The VoiceXML, as an essential part in the dialog manager, will create and manage all possible dialogs and their logical relationships.

4.2 Development Methodology

To achieve the requirements of this mobile voice-based application, the project follows the Iterative Incremental Development (IID) methodology, as shown in Figure 22. The IID methodology is viewed as a combination of iterative development and incremental development focusing more on development phase in the process.

FIGURE 22 DEVELOPMENT METHODOLOGY: ITERATIVE INCREMENTAL DEVELOPMENT

The figure 22 shows that the whole development process is separated into 3 parts: System analysis and design, incremental development and evaluation. In the system analysis and design phase, project should analyse all main and optional requirements and accomplish design related tasks, such as building system architecture, designing GUI and designing VoiceXML files. Meanwhile, it is required to predict possible risks during the development phase, and consider corresponding solutions. In the development phase, this project creates a prototype for application before starting incremental development. Each development unit will be implemented and tested until the application caters for all requirements proposed in design phase. Once the final version of software is finished, it will be evaluated in evaluation phase. Following section will introduce details of evaluation methodology.

4.3 Evaluation Methodology

This section will introduce how to evaluate the hypothesis that “Using proposed mobile emergency application is faster and better to provide aid than using existing web-based first-aid websites” mentioned above. Walker [35] provided a typical evaluation framework (named PARADISE) to test spoken dialog agents. This framework combines two main features comparing different dialog agents, such as success (k) and dialog cost (C) [35]. Success (k) is used to rate whether voice agents could reach their strategies[35]. For example, in “El Siri” application I proposed, the main strategies are diagnosing emergency,
checking severity of the problem and providing corresponding first aid advice. On the other hand, the feature of dialog cost ($C$) could be calculated based on the dialog behaviours of end users and system [35], such as the number of turns and operating time in this “EI Siri”.

According to PARADISE [35], I design an experiment to prove the hypothesis related to “EI Siri” mentioned before. In this experiment, I will find 20 testers to try my application and compare it with another way (e.g. the combination of “Esagil.org” website [36] and “British Red Cross” application [31]). They should fill an information form before experiment, as shown in Figure 23.

![Evaluation Questionnaire for Intelligent Emergency Assistant](image)

**FIGURE 23 INFORMATION FORM IN EVALUATION**

Figure above shows that there are some details about testers’ knowledge about mobile application and voice interface, as well as testers’ native language. Since these factors could affect the results of evaluation, I will respectively analyse all experiential data under different cofactors in evaluation phase. Moreover, I will provide 4 main different case cards, which contain different situations and relevant symptoms, as shown in Figure 25.

![Case Card](image)

(a) Case Card 1: Burnt

![Case Card](image)

(b) Case Card 2: Unclear Situation – Food Poisoning
In these figures, all case cards are divided into two parts, including clear situation (such as burns and sprains) and unclear situation (such as asthma and food poisoning). 20 testers in this experiment are required to carefully read different case cards and to find corresponding first aid advice in two different ways. They also need to answer a set of relevant questions once they find corresponding medical suggestions. A questionnaire is designed in the following Figure 25.

According to this figure, all testers could answer several simple questions after each case. I will record the operating time for each case. At the end of evaluation phase, I will gather and analyse all data to prove the hypothesis, including users’ answers and the operating time. To be similar with PARADISE [35], this time, as the dialog cost, will be combined with testers’ answers and analysed for final results. It could ensure whether the proposed mobile voice application (“EI Siri”) is more effective than the existing applications or not.


**Requirement Analysis**

5.1 Overview

This section will list and analyse all requirements related to “El Siri” I proposed in this project. They need to be considered and implemented to support hypothesis in this project. The main purpose of this project is to build a mobile voice application for emergency. It focuses on helping people in an emergency. This application needs to confirm the potential situation, analyse the severity of problem and rescue patients. These steps will play important roles in life-saving. This mobile application will also provide corresponding solutions to patients or first responders, such as providing a simple aid advice or contacting Emergency Centre.

All requirements will be separated into two parts: user requirements and system requirements. User requirements means what end-users need to achieve from software. And system requirements contain all features related to mobile application. Both of them will be shown in the following sections.

5.2 User Requirements

In order to improve the satisfaction of users, there are several requirements (as shown in Table 2) which are meaningful for end users. They are discussed about what end users want to achieve from “El Siri” I proposed. More details will be analysed below.

<table>
<thead>
<tr>
<th>No.</th>
<th>Requirements</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Achieving a natural dialog</td>
<td>High</td>
</tr>
<tr>
<td>2</td>
<td>Receiving a final result for diagnosis and first aid</td>
<td>Medium</td>
</tr>
<tr>
<td>3</td>
<td>Getting shorter operation time (less than 5 minutes)</td>
<td>Medium</td>
</tr>
<tr>
<td>4</td>
<td>Getting simple operation mode</td>
<td>Low</td>
</tr>
</tbody>
</table>

*Table 2 User Requirements for “Emergency Intelligent Siri”*

Achieving a natural dialog

Obviously, interaction between human and device is the easier, the higher efficiency of application will be. Users want to communicate with mobile application by a natural dialog rather than by remembering a set of complex commands, when they use it. It will be more convenient for users to get accurate results as soon as possible. I will develop a particular dialog manager with VoiceXML technique to build a suitable dialog environment to fulfil this requirement.
Receiving a final result for diagnosis and first aid

It is impossible for users to understand and remember complex steps of first aid for patients especially in emergency. Corresponding images, speech or text could enable them to be easier to understand what happens and how to provide first aid after conversations. Users should be aware of a result about the situation and relevant aid information from system in the form of text or image. The result will contain the situation of emergency, severity, and corresponding rescue measures. It also includes the detailed location of emergency, which could show where they are and help them to contact with Emergency Centre. All details mentioned before are recorded in a template file on SD card, which will be explained in the section of system requirements. There is an example about final result in the form of text, as shown in Figure 26.

![Figure 26 Example of Final Result in the Form of Text in Application](image)

Getting shorter operation time (less than 5 minutes)

As mentioned above, users or first responders hope to get final results about emergency situation and relevant aid information as soon as possible. The less time users could use, the greater the chance of salvation is. In order to meet this need, the whole dialog should be limited less than 5 minutes. Hence, they could spend much more time on saving life.

Getting simple operation mode

To be similar with requirement of getting shorter operation time, end-users prefer to spend much more time on getting the diagnosis results and providing first aid than on studying how to use this application. Simpler operation mode could make this emergency application to be easier to use in an emergency. The number of screens in the final application should
be less than 3, and all screens will be designed as simple as possible, such as an example of voice mode in application in Figure 27.

![Figure 27 Example of Voice Mode in Application](image)

**5.3 System Requirements**

In order to improve system, this section lists a set of features and functionalities of “Emergency Intelligent (EI) Siri”, as shown in Table 3. More details are discussed below.

<table>
<thead>
<tr>
<th>No.</th>
<th>Requirements</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Developing application based on Android platform</td>
<td>High</td>
</tr>
<tr>
<td>2</td>
<td>Avoiding limitations of Internet</td>
<td>High</td>
</tr>
<tr>
<td>3</td>
<td>Recognising and responding voice information using Google APIs</td>
<td>High</td>
</tr>
<tr>
<td>4</td>
<td>Focusing on 5 main emergency situations</td>
<td>Medium</td>
</tr>
<tr>
<td>5</td>
<td>Building an application without login function</td>
<td>Medium</td>
</tr>
<tr>
<td>6</td>
<td>Building dialogs based on system-initiative mechanism</td>
<td>High</td>
</tr>
<tr>
<td>7</td>
<td>Analysing dialog content using VoiceXML technique</td>
<td>High</td>
</tr>
<tr>
<td>8</td>
<td>Locating user's position using Google API</td>
<td>Medium</td>
</tr>
<tr>
<td>9</td>
<td>Recording all emergency related information</td>
<td>Medium</td>
</tr>
<tr>
<td>10</td>
<td>Calls should be logged for such as start-time and end-timer</td>
<td>High</td>
</tr>
</tbody>
</table>

*Table 3 System Requirements for “Emergency Intelligent Siri”*

**Developing application based on Android platform**

Currently, mobile devices have been essential in people's daily life. Among one of the most popular platform is Android, which occupy nearly 70% of global market, more than that of
Apple platform (about 20%). On the other hand, Google Company provides a set of APIs supporting various functions or features to android developers. Hence, voice application in this project will be developed based on Android platform. I will implement this application by using Eclipse IDE, which is a main Java compiler.

Avoiding limitations of Internet

This application is proposed to help patients in emergency. It needs to check emergency situation, level and provide aid as soon as possible. However, performance is probably affected by other factors of environment, such as Internet. Application needs to slow down or crash directly once Internet is with a bad quality or cannot be found. In order to cope with this problem, I will develop it with a stand-alone version. All processes will be performed on mobile devices. Moreover, system will check Internet in the beginning of application. If Internet is good, users are allowed to provide inputs in the form of speech. But users will be required to enter text inputs instead of voice when Internet is bad.

Recognising and responding voice information using Google APIs

Google APIs such as speech API and text-to-speech API are really useful for developing spoken dialog systems. Both of them provide all interfaces and methods to process voice inputs and generate voice outputs respectively in this application. I will use them to implement the speech-input and speech-output components of mobile voice application.

Focusing on 5 main emergency situations

As a mobile voice application in emergency, there could be a significant number of emergency situations in the real world. It seems to be difficult to consider all situations in this project. Therefore, this application in the project will focus on 5 main situations, including burns and scalds, sprains and strains, heart attack, food poisoning as well as asthma attack. They might be divided into two parts, one has specific statements about situations (e.g. burns and scald) and another one needs to be analysed via a set of question during this application (e.g. heart attack and asthma attack)

Building an application without login function

In the emergency, time is really important especially when saving life. No one wants to spend any time on register or log into systems in emergency. Voice application in this project will be free for all users. It could not have the function of login or registration. Users could diagnose situations of emergency and get first aid directly without any redundant steps.
Building dialogs based on system-initiative mechanism

Users could not precisely describe the patient’s state in the emergency because of tension or lack of medical knowledge. The diagnosis part is the core of this voice application. System-initiative mechanism in diagnosis part could help application analyse and confirm patient’s state as soon as possible as shown the following example in Figure 28.

```
Clinic: Hi. Can I help you?
User: I found a guy in trouble
Clinic: Okay, I need you to answer a set of questions to confirm the situation. Just tell me yes or no please.
Is the patient hard to breathe?
User: no
Clinic: Does patient has vomiting?
User: yes
Clinic: Dose the injured area look crooked or has lumps or bumps?
User: yes
Clinic: Does patient has diarrhoea?
User: yes
Clinic: Does patient has stomach cramps?
User: yes
Clinic: This patient may have a food poisoning.
```

**FIGURE 28 EXAMPLE OF SYSTEM-INITIATIVE MECHANISM**

In the figure 28, users only need to provide simple answers (“yes” or “no”) for questions of mobile system. Application could confirm the real situation via a set of logical relations, such as the conditions of heart attack, food poisoning as well as asthma attack.

Analysing dialog content using VoiceXML technique

VoiceXML technique could contribute to build an excellent dialog network, in which it shows logical relationships between inputs and outputs from system and users. Robust VoiceXML could support a natural and continuous dialog and provide enough evidences to systems to analyse dialog content. In this project, VoiceXML is viewed as an essential component building emergency diagnosis mechanism.

Locating user’s position using Google API

Google Location API could use GPS function to help application to find out specific position of emergency, which is meaningful especially when contacting Emergency Centre. Accurate information enables aid workers to arrive at emergency location and aid patients on time. Voice application proposed will implement location functions using Google location API in the development.
Recording emergency related information

Obviously, asking duplicate or additional questions about the same situation is really important in an intelligent dialog system. The majority of modern voice applications only answer questions or perform tasks provided by users rather than build a logical relation networks. It means that systems cannot remember any details about what they mentioned in the last dialog. For example, Siri focuses on answering question from users and helping users to search for required information, but not analysing logical correctness or relationships of users’ statements. Template recording file could remind application to check information about current situation.

Application could create a template file for recording basic information about emergency, including the location of users or emergency, the state of patients and detailed date, when first responders contact Emergency Centre. Otherwise, once users may request a little repeat of medical solutions about the emergency, the recording could also help system to present detailed aid. Template file with all information will be stored in a particular folder on SD cards.

Calls should be logged for start-time and end-timer

At the end of project, mobile voice application will be evaluate with real data. In order to test this mobile application effectively, it seems to be really necessary to record the state of calling diagnosis, especially the start-time and end-time. In this system, it required to get specific time of each phase, such as diagnosis part, confirmation part or first aid part. They will reflect the speed or efficiency of mobile application I proposed in a given emergency.

5.4 Optional Requirements

Apart from these main requirements introduced above, there will be several optional requirements improving the mobile voice application. They will provide further influence on the future of this application. The optional requirements are an interface for dynamically adding new emergency situations, repeating first aid, contacting Emergency Centre directly, as described below.

An interface of adding new emergency situations

In order to extend this intelligent emergency assistant application, this proposed system need to provide an interface for new emergency situations. Developers could dynamically add new emergency situations, and related information (e.g. aid information and VoiceXML) into this application through existed interface.
Repeating first aid

Sometime users or first responders may request repeating the first aid or related information because of tension or lack of relevant knowledge. System could receive new commands and provide corresponding aid in the form of voice according to the template recoding file mentioned before.

Contacting Emergency Centre directly

System could help users contacting Emergency Centre directly, so that users could concentrate on helping patient who is in emergency. Application needs to automatically provide all emergency related information from template recoding file to Emergency Centre and tell users when the medical officers could be arrive at emergency location.
PROFESSIONAL, LEGAL AND ETHICAL ISSUES

6.1 Privacy Issues

This application should consider the privacy of users, because some users or patients do not want to share their information (e.g. personal information or illnesses) with others. Emergency application in this project will be developed based on an open-source mechanism. Users do not need to register and sign into systems before using it. System will remove all documents about users' personal information and situation of problems from mobile phones. It could prevent users' privacy leakage.

6.2 Safety in Emergency

Mobile voice application proposed in this project needs to provide some suggestions about first aid to users or first responders. These advices are related to save human's life or eliminate the possible risks. However, it is difficult to ensure all of aid suggestions are accurate and suitable for patients in emergency. Misleading on emergency aspect might lead to an extremely dangerous situation, in which patients might aggravate their conditions and even lose their life. This application will design a message of statement about that "All suggestions about first aid are just your references to help your friends. Please contact and ask advices from professionals." System will send it to users before starting to diagnosis.

6.3 Ethical Issues

Voice application is designed to help users to contact Emergency Centre when the severity of the problem is high. In the evaluation phase, I will assume a set of emergency situations for testers. All participants should diagnose given situations and find corresponding medical aid. In order to prevent affecting the Emergency Centre, proposed application will stop the function of contacting the Emergency Centre while providing first aid advice. All participants will be announced that all emergency situations in the evaluation phase are hypotheses. There also needs to be an informed consent. All testers must sign a form for permission to use data. On the other hand, mobile application I proposed is an emergency suggestion application but not a medical device, due to the fact that all information about illness diagnosis and first aid are collected from the websites of “NHS Choices” [1] and “British Red Cross” [2]. It could not ensure all information are significant correct and helpful for patients in emergency. Inaccurate diagnosis or help would affect the patients in the rescue process. Hence, there should be a disclaimer that “All information and activity for emergency rescue are for medical suggestion only. Users are encouraged to confirm the information with other sources or contact with Emergency Centre (999) directly.”
**PROJECT PLAN**

### 7.1 Stakeholders and Tasks

#### 7.1.1 Stakeholders

This project aims to propose an intelligent emergency assistant based on a voice interface. This application will be supported on Android platform. The voice-based application could help users diagnose the conditions of patient who is in the emergency, contact emergency centre and teach users how to provide first aid. Therefore, there are several stakeholders in the project, including users or first responders, patients, emergency centre. Effective diagnosis could successfully save patients’ life or minimise potential risks of emergency situations. Clear guidance could assist users or first responders to provide first aid. Emergency Centre could benefit from the application with high quality and speed.

#### 7.1.2 Tasks

The whole project will be separated into four phases. In the first stage, named design stage, I will learn voice-application related techniques (e.g. Android platform, Google APIs and VoiceXML) as well as design VoiceXML files for next stage. I also need to analyse all requirements and design a set of relevant UML diagrams for application. The second Stage is called implementation stage, in which I will build the prototype of application and improve it incrementally. Stage two plays an essential role in the whole project. The third stage will focus on evaluation of developed application and analysis of results from evaluation. Finally, I will combine different parts of report to finish the final version, submit it with my final software and prepare for my poster at the end of project.

### 7.2 Project Schedule

The following figure 29 shows the Gantt chart about this project. In the diagram, there are a timeline for all tasks in project development. It consists of design stage, implementation stage, evaluation stage as well as report-writing stage. The design stage includes study of Google APIs, analysis of requirements, and some design tasks for UML diagrams and VoiceXML. Writing design part of report also belongs to this stage. Implementation stage includes building prototype, developing GUIs, develop dialog manager and location tracker, linking all components as well as all relevant software tests. Stage two also contains implementation report writing. In evaluation stage, there are designing questionnaires, application evaluation, analysis of results, and evaluation report writing. The final stage (report stage) only contains report writing and updating, and poster preparation.
7.3 Risks

Although the project schedule described above has considered more details about mobile voice interface project, there may be also some risks. This schedule does not contain the rest time and meeting time. Moderate rest could improve efficiency of working in development. Meeting with supervisor could find out potential problems during the development and get more useful information about project.

On the other hand, this schedule also omits some potential risks I cannot predict now. They are listed with their likelihoods in the following Table 4.

<table>
<thead>
<tr>
<th>Risks</th>
<th>Importance</th>
<th>Likelihood</th>
<th>Back-up Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech recognition is not accurate enough</td>
<td>High</td>
<td>Medium</td>
<td>Using the touch screen or GUI instead</td>
</tr>
<tr>
<td>VoiceXML Interpreter cannot be effective in system development</td>
<td>High</td>
<td>High</td>
<td>Using own interpreter to analysis XML files instead of VoiceXML</td>
</tr>
<tr>
<td>Location service failure</td>
<td>Medium</td>
<td>Medium</td>
<td>Remove location function from requirements list</td>
</tr>
<tr>
<td>Inefficient to diagnose the emergency situation</td>
<td>Low</td>
<td>Low</td>
<td>Reorganizing the VoiceXML in system or change situation to another one</td>
</tr>
<tr>
<td>System cannot provide aid</td>
<td>Low</td>
<td>Medium</td>
<td>only provide first aid in the</td>
</tr>
<tr>
<td>related images or text</td>
<td>form of voice</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intrusion of background voice in the speech recognition</td>
<td>Medium High</td>
<td>Using the touch screen or GUI instead</td>
<td></td>
</tr>
<tr>
<td>Application crash</td>
<td>Medium Low</td>
<td>Using SVN to manage different version of application</td>
<td></td>
</tr>
<tr>
<td>System cannot provide an interface for new emergency information</td>
<td>Low Low</td>
<td>Just provide first aid in form of audio</td>
<td></td>
</tr>
<tr>
<td>Testers in evaluation cannot automatically log operation time accurately</td>
<td>Medium Medium</td>
<td>Changing evaluation questionnaire and focusing more on efficiency of application and feedbacks.</td>
<td></td>
</tr>
<tr>
<td>Requirements temporary change</td>
<td>Medium Medium</td>
<td>Analysing new requirements and order them with ns priorities</td>
<td></td>
</tr>
<tr>
<td>Student cannot meet supervisor timely</td>
<td>Medium Low</td>
<td>Make an appointment online, such as Skype or MSN.</td>
<td></td>
</tr>
<tr>
<td>Testing mobile crash</td>
<td>Low Low</td>
<td>Borrowing another mobile phones from friends</td>
<td></td>
</tr>
</tbody>
</table>

*TABLE 4 RISKS IN DEVELOPMENT OF MOBILE VOICE INTERFACE PROJECT*  
(H: High M: Medium L: Low)
SYSTEM DESIGN

8.1 OVERVIEW

The “Emergency Intelligent (EI) Siri” application on mobile devices should be designed with feasibility and usability in terms of system execution. It should also be easy and friendly for end-users in emergency and developers in improvement. The main purpose of this chapter is to discuss about system design and what potential problems I considered during this stage. Moreover, I will talk about what differences or improvements there will be in system design in order to achieve the goal of project.

8.2 ARCHITECTURE DESIGN

Since this project aims at developing a voice-based interface on mobile devices for emergency, it is required to analyse emergency situation and its level, as well as to provide corresponding rescue measures. According to requirements I analysed, this application should avoid the limitation of Internet. Hence, it could have a good performance no matter whether there is an Internet connection with good quality or not. The “Emergency Intelligent (EI) Siri” is a kind of standalone software, which means that most of processes in application (e.g. dialog manager and text-to-speech) are performed on mobile devices. However, speech recognition is developed based on Google server, as shown in Figure 30.

![Figure 30: Basic Architecture of “EI Siri”](image)

This figure shows that the whole application is based on Android version 4.0 and developed with Google APIs. End users are allowed to provide speech inputs and receive speech outputs respectively from the application. All voice messages will be converted to text and processed in the dialog manager, which is the core of generating a natural dialog in an
A Mobile Speech Interface for Emergency Help: EI Siri

Yanchao Yu (H00099161)

Responses from the dialog manager may be converted back to speech for users at the end of each turn. This application is an open system, which means that users could use it without user-name and password. This could shorten the execution time.

Although Internet is not really indispensable, the phase of speech recognition in “EI Siri” is developed by Google Speech API relied on Internet. Therefore, I will provide an additional way to enter text inputs. The system could require users to change to manual mode depending on the status of Internet. The entire activity of the application is shown below (Figure 31).

**FIGURE 31 ACTIVITY DIAGRAM OF “EMERGENCY INTELLIGENT SIRI”**

In figure 32, the system will create a system folder and files when users start up application for the first time. It could also record the emergency date and related location in the...
beginning each time. Users could use voice mode to provide speech inputs if there is Internet. Otherwise, they are required to check Internet settings or enter manual mode directly. During the emergency dialog, users need to provide a natural sentence to system. All inputs are analysed in the dialog manager, and answer are returned to users. This figure also points out that there is a limitation for speech recognition, which means that users are informed to change input mode once the number of errors is more than 3. At the end of the application, users receive a final text result, which contains all details about current emergency, such as emergency situation, level, and first aid information.

8.3 Functionality Design

8.3.1 Use Case Modelling

In the “Emergency Intelligent (EI) Siri” application, there are several main activities with users, as shown in Figure 32. They could start up a new diagnosis process by clicking the “start” button and are allowed to change the mode of input. Users will perform a natural dialog for emergency and check text results after it.

![FIGURE 32 USE CASE DIAGRAM ON “EMERGENCY INTELLIGENT SIRI”](image)

Figure 33 shows that there are two basic activities (including providing speech input and receiving speech output) in the activity of performing a natural dialog. I will develop the functionalities of speech input and speech output using an API of “Speech Recognition” and an API of “Text-To-Speech” respectively provided by Google. There is a text description for details relating to Use Case diagram, as shown in Table 5.
## Text Description for Users (first responders or patients)

<table>
<thead>
<tr>
<th><strong>USE CASE</strong></th>
<th>Emergency Intelligent Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GOAL</strong></td>
<td>To diagnose emergency and provide corresponding aid</td>
</tr>
<tr>
<td><strong>ACTOR</strong></td>
<td>User, El Siri</td>
</tr>
</tbody>
</table>

**MAIN SUCCESS SCENARIO:**
1. User starts up emergency diagnosis
2. System check Internet
3. User provides speech input to application
4. System analyses inputs and search for responds
5. User change input mode
6. Finish emergency dialog
7. User reads final text result

**ALTERNATIVE:**
1a. Cannot find SD card
   1. Provide warning about “Cannot find SD card”
   2. Exit application
1b. there are not system files
   1. Create system folder on SD card
   2. Create template XML file in system folder
2a. Internet cannot be connected
   1. Provide warning dialog to enter manual mode
4a. Cannot find keywords in inputs
   1. Add error number
   2. Require user to change input mode to manual one
6a. System find symbol of end of dialog
   1. Close TTS object
   2. Change activity to end

### TABLE 5 TEXT DESCRIPTION FOR APPLICATION

### 8.3.2 Structure modelling

Structure modelling is viewed as a kind of formal method to represent different objects and associations between them. It could also describe and explain use case diagram (which mentioned in the previous section) in detail. There is a class diagram for “El Siri” application, which could list all main activities and functions related to application, and link them to each other, as shown in Figure 33.
A Mobile Speech Interface for Emergency Help: EI Siri

Yanchao Yu (H00099161)

The figure above shows a set of main classes on this application, including Splash, EIMainActivity, EIClinic, EIEnd as well as several functional classes. Their functions are explained below:

1) The class of Splash is used to show the splash screen in the beginning of application and create folder and files relating to system. This class also records emergency date and specific location in template XML file. The functional class of GPSTracker in class diagram could help the system to find longitude and latitude of emergency.

2) In EIMainActivity class, the application could check the status of Internet when users start up a new emergency diagnosis dialog.

3) EIClinic class is responsible for generating a natural dialog between user and the mobile application. It will process inputs from users in the dialog manager and respond to users in the form of speech. There are two different modes in one activity, including voice mode and manual mode. Voice mode may allow users to provide audio input to the
dialog manager, while manual mode will generate a dialog via text input. On the other
hand, Classes of MessageAdapter and NewMessageItem are designed to list user inputs
and responds in EIClinic activity.

4) At the end of the application, the system will provide a text result to users in the class of
EIEnd. Users also try to restart the application by clicking “restart” button.

5) TempLog, as a functional class, is developed with Singleton pattern. It could record all
information relating to the emergency (e.g. emergency situation, level of illness and
rescuing measures) into template XML file, which is created automatically in the
beginning of application.

Moreover, other classes are designed to build a completed dialog manager in this
emergency application. They will be described and explained in the next section.

8.3.3 Dialog Manager

Dialog Manager (DM), as the core of “Emergency Intelligent Siri” application, aims at
analysing inputs from users and generating corresponding output in the form of text. It
should be built to process several voice-based files (e.g. VoiceXML files) to create a natural
conversation with users. Traditionally, many voice applications tend to develop their own
dialog manager using a VoiceXML interpreter, which is developed to parse VoiceXML files
and explain their interrelations. VoiceXML interpreter could quickly search for responses via
those grammars in files. Currently, there are several main VoiceXML Interpreters in the
development of voice-based applications, such as OpenVXML and JVoiceXML.

Unfortunately, because the majority of VoiceXML interpreters (e.g. JVoiceXML) are open
source on websites, their authors will stepwise develop them and cannot provide as clear
user guides as commercial software provides. It makes it difficult for developers to develop
their own voice applications. Moreover, some interpreters have integrated dialog manager
with functionalities of input and output in the form of speech, but not text. In my design of
application, I develop speech recognition and text-to-speech with Google APIs. To compare
with other interpreters, although JVoiceXML allows developers to develop their own speech
interfaces, it cannot support Android development. Hence, I decide to build an independent
dialog manager.

In my dialog manager, I processed a set of xml files instead of VoiceXML files. These xml files
will follow the logical structure and interrelations of traditional VoiceXML without complex
grammars. They will be explained in the next section. In order to process these xml files, I
design a structure model (so-called class diagram) in Figure 34.
This figure shows that dialog manager I proposed contains a dialog interface and four classes. The dialog interface, named "DManager", will declare a set of methods to start up sub-dialogs, search and return responses, as well as find next node in a dialog. All classes in dialog manager will implement this interface to generate sub-dialogs, such as situation analysis, level analysis and first aid.

Classes in dialog manager are "DialogManager", "SituationAnalysis", "LevelAnalysis" and "FirstAid". I may introduce them below:

1) The class of "DialogManager" aims at checking specific situation in dialog (e.g. food poisoning and heart attack) or unclear problems (e.g. ill, sick and emergency).
2) The class of “SituationAnalysis” is proposed to confirm situation using a set of questions. It focuses on food poisoning, heart attack and asthma attack. Users just need to provide ‘yes’ or ‘no’ to answer these questions in this phase.

3) “LevelAnalysis” class is designed to check the level of illness by asking some questions. If there are not conditions found in this phase, illness will be slight, otherwise it should be severe.

4) “FirstAid” class will provide a set of rescue measures for particular emergency. The system could contact Emergency Centre directly via SMS message once the level of illness is severe. If not, the system may only provide corresponding aid information to users.

Additionally, the system needs to provide an error statement when text input cannot be matched in dialog. Details (e.g. algorithm and code) will be described in the chapter of implementation.

8.3.4 VOICE-BASED XML FILES

In the design of dialog manager, a set of voice-based files play important roles. They are designed to generate a natural dialog. I designed several XML files simulating VoiceXML files, which flow the similar written rules in VoiceXML file.

Basic Structure

In this mobile application, I designed a general structure of voice-based XML files, which includes 4 main XML files in Figure 35.

**FIGURE 35 XML FILES STRUCTURE IN “EMERGENCY INTELLIGENT SIRI”**
As shown in Figure 3, there are 4 main XML files for emergency dialog, including general file, situation file, level file as well as first aid file. General XML file focuses on the choice of emergency situations and the end of dialog. The system will jump to next file according to different situations via keywords. The situation File is designed to support situation diagnosis, which contains a set of questions. The system needs to check the specific situation via users’ answers. The level file could be activated to analyse the level of illness after the situation is checked. There will be 4 main sub-dials, including food poisoning, asthma, heart, Burns and scalds as well as Sprains and strains. The situation of heart attack will jump to rescue file directly once it is confirmed, because it is really severe. In each sub-dialog, the system could analyse all questions to check the level.

On the other hand, the system may decide to how to help patients in emergency when the level is found. According to the figure above, system needs to contact the Emergency Centre directly and provide rescue measures once the level of illness is severe. Unless, it could only help users to deal with problems based on known situation.

Format of Voice-based XML File

In terms of the format of the voice-based XML file, there will be a set of common attributes in each node. They are introduced below:

1) The attribute of “value” represents the id of node, such as “yes” or “no” in the node of “answer”.
2) The attribute of “para” represents the parameter used in application, such as “location”
3) The attribute of “statement” represents what will be responded in speech. It will be transferred to mobile activity for speech output.
4) The attribute of “next” points to next node in XML files, which system could process in the next step.

Unified format of XML file could improve the efficiency in the development of application and enable system to be updated easily in the future.

XML File Design

In the basic architecture mentioned before, there are several sub-dials in each XML file. They will be designed to manage different steps in emergency diagnosis. I will introduce them below.

General file, as the root of dialog tree, is created to manage the beginning and end of dialog. There is a dialog tree for this file, as shown in Figure 36.
A Mobile Speech Interface for Emergency Help: EI Siri

Yanchao Yu (H00099161)

FIGURE 36 DIALOG TREE OF GENERAL XML FILE

This figure shows that this file will provide all possible keywords (e.g. emergency, burned and food poisoning) to start a new dialog in the “start” sub-dialog. It also provides a statement when completing dialog at the end.

The file of situation-analysis provides a few questions to diagnose a specific situation. It focuses on three main emergency, including heart attack, asthma attack as well as food poisoning. System will get situation statement and provide next node in dialog by providing keywords of “yes” and “no”, as shown in Figure 37.

FIGURE 37 DIALOG TREE OF SITUATION-ANALYSIS XML FILE [1], [2]

In level analysis stage, there is an xml file containing several sub-dialogs, including asthma attack, food poisoning, burns and scalds, as well as sprains and strains. Each sub-dialog will provide several conditions for checking illness level. There are a set of dialog trees in Figure 38.
According to this figure above, once answers from users could be matched to one of these conditions, this situation will be severe, otherwise it could be slight.

First-Aid XML file is designed to provide rescue advices for particular situation depending on severity of problem. It will be required to contact Emergency Centre and provide several first aid suggestions, as shown in dialog tree of XML file in Figure 39.
8.4 User Interface (UI) Design

This section introduces the user interface (UI) design of “Emergency Intelligent Siri” application. I will provide information relating to guidelines of GUI design and layout of UI in this proposed application.

8.4.1 Eight Golden Rules of Interface Design

This section will list the “8 Golden Rules of Interface Design” by Ben Shneiderman [37], which improve the usability of “El Siri” on the design aspect (as shown in Table 6). These rules are significantly useful and helpful in designing a user-friendly interface [37].

1. Strive for Consistency

“Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent commands should be employed throughout.”

2. Cater to Universal Usability
“As the frequency of use increases, so do the user's desires to reduce the number of interactions and to increase the pace of interaction. Abbreviations, function keys, hidden commands, and macro facilities are very helpful to an expert user.”

3. **Offer Informative feedback**

“For every operator action, there should be some system feedback. For frequent and minor actions, the response can be modest, while for infrequent and major actions, the response should be more substantial.”

4. **Design Dialogs to yield closure**

“Sequences of actions should be organized into groups with a beginning, middle and end. The informative feedback at the completion of a group of actions gives the operators the satisfaction of accomplishment, a sense of relief, the signal to drop contingency plans & options from their minds, and an indication that the way is clear to prepare for the next group of actions.”

5. **Prevent Errors**

“As much as possible, design the system so the user cannot make a serious error. If an error is made, the system should be able to detect the error and offer simple comprehensible mechanisms for handling the error.”

6. **Permit easy reversal of actions**

“This feature relieves anxiety, since the user knows that errors can be undone; it thus encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data entry or a complete group of actions.”

7. **Support internal locus of control**

“Experienced operators strongly desire the sense that they are in charge of the system and that the system responds to their actions. Design the system to make users the initiators of actions rather than the responders.”

8. **Reduce short term memory**

“The limitation of human information processing in short-term memory requires that displays be kept simple, multiple page displays be consolidated, window-motion frequency be reduced and sufficient training time be allotted for codes, mnemonics and sequences of actions.”

**TABLE 6 Eight Golden Rules of Interface Design [37]**

### 8.4.2 Screen Layout

According to the requirements in proposed application, user inter there is a screen layout for mobile application, as shown in Figure 40.
8.4.3 Main Screen Design

In this section, I will describe the main screen designs with voice mode and manual mode, which could provide a natural interaction between users and device. It contains functional modes (such as voice-input and text-input) as well as relationships between them.

In the voice-mode of the application (as shown in Figure 41), end-users could activate the speech input mode from Google Speech API by clicking “Voice” button on the clinic screen. System will show inputs from users and responds from dialog manager in the form of text in a list. Therefore, users could know all details in an emergency dialog. On the other hand, users could choose to change it to manual mode (which will be described below) by clicking “manual” button.

![Figure 40: Screen Layout of Mobile Application](image)

**FIGURE 40 SCREEN LAYOUT OF MOBILE APPLICATION**

![Figure 41: Voice Mode Screen](image)

**FIGURE 41 VOICE MODE SCREEN**
To be similar to voice mode (as shown in Figure 42), user need to type text inputs into an EditText and send it to dialog manager via “Send” button. This screen could only list text responds from server. However, users cannot be allowed to return back to voice mode once this mode is activated.

![Figure 42 Manual Mode Screen](image)

**FIGURE 42 MANUAL MODE SCREEN**

After an emergency dialog, a result screen is activated to show final text result (as shown in Figure 43). The result will include information relating to emergency situation, such as date, emergency, level of illness, aid measure as well as start and end time. Users are also allowed to restart application via a button of “restart”.

![Figure 43 Result Screen](image)

**FIGURE 43 RESULT SCREEN**
9.1 Overview

In this chapter, I will describe and explain the implementation stage of the project. The whole application is developed depending on the Iterative Incremental Development (IID) methodology. Hence, there are several main stages, including system preparation, prototyping, core implementation as well as GUI implementation.

In system preparation, I made a configuration for application and prepare all Android xml files and resources. In prototyping, I developed two main functionalities of speech input and output. It also contains an important function of tracking location. The stage of core implementation designed and developed a dialog manager, which aims at building a natural dialog for emergency help. Finally, I created all user interfaces for application in this stage of GUI implementation. All details will be explained in the following sections.

9.2 System Preparation

In order to implement “EI Siri”, there are a significant number of configurations considered before or during this stage, such as the version of Android operating system, the number of screens, as well as different user permissions. All of these properties are configured in a XML file of “AndroidManifest”.

According to the tutorial of Google API, Text-To-Speech has different ways to be implemented according to the version of Android. It means that when the version of Android is lower than Android 4.0.3, system will use a kind of function to control speech generation. While the version is higher, there will be another way to implement it. I determined to develop the “EI Siri” under Android 4.0.3(level 14), because of the version of my testing mobile, as shown a snippet in Figure 44.

```xml
<uses-sdk
    android:minSdkVersion="10"
    android:targetSdkVersion="14" />
```

**Figure 44** Version of Android in AndroidManifest.xml

Due to the user requirements mentioned before, system is required to control the number of screens or activities, which should be less than 5 screens. I decided to build 4 main activities, including Splash, EIMainActivity, EIClinic and EIEnd, as shown in Figure 45. They will be described in the following sections.
Moreover, there are a few functionalities developed to meet different needs of users and system, such as tracking location, searching speech results online, contacting Emergency Centre as well as accessing to files on SD card. In order to implement them, some permission properties are declared in AndroidManifest.xml (Figure 46).

On the other hand, in order to beautify the GUIs in application, I designed a set of pictures and button animations stored in a “drawable” folder. And XML files in this folder are used to define button images in different status, such as start button shown in Figure 47.
9.3 PROTOTYPING

The prototype of application consists of three main functionalities, including speech recognition, Text-To-Speech as well as location tracker. They are developed to build a simple dialog with single turn. I will describe them separately in the following.

9.3.1 SPEECH RECOGNITION

Speech recognition is developed based on Google Speech API, which focuses on converting audio input to text. System could create a new instance of the speech recognizer with a parameter of SPEECH_R to collect voice data on mobile devices shown in Figure 48.

```java
Intent listener_intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
listener_intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE, getClass().getPackage().getName());
listener_intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Waiting for Voice...");
listener_intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
listener_intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 10);
startActivityForResult(listener_intent, SPEECH_R);
```

**FIGURE 48 INITIALLING SPEECH RECOGNITION**

The parameter of SPEECH_R initialled in the beginning of the activity plays an important role in the process of speech recognition. Once the value of request code is equal to SPEECH_R, the system will search all potential matched results and store them in an array list. All results are sorted in the matching index. I required the system to find the most matched result, which will be used for analysis in the dialog manager. There is a snippet shown in Figure 49.

```java
if(requestCode == SPEECH_R && resultCode == RESULT_OK )
{
    ArrayList<String> result_txt = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
    _input = result_txt.get(0);
```

**FIGURE 49 COLLECTING VOICE DATA**

9.3.2 TEXT-TO-SPEECH

In contrast to the speech recognition, Text-To-Speech is developed to convert text to voice. I developed 3 main methods including tts_Speak, onInit as well as onUtteranceCompleted. The method of tts_Speak creates a hash map to store text stream and Utterance ID in Text-To-Speech. The system could automatically convert text to speech using speak method (as shown in Figure 50).
The method of `onInit` is inherited from an interface of `OnInitListener`. When an object of Text-To-Speech is successfully created, it could set the language mode (the language in application is UK English). On the other hand, this method also starts up the dialog manager and converts a response to voice. It is shown in Figure 51.

```java
if (status == TextToSpeech.SUCCESS) {
    tts_emergency.setLanguage(Locale.UK); 
    tts_emergency.setOnUtteranceCompletedListener(this); 
    
    this._output = checkResponds(dialog_manager.startUp("start")).trim(); 
}
```

**FIGURE 51 INITIALLING TEXT-TO-SPEECH**

The method of `onUtteranceCompleted` plays an essential role in building a natural dialog. It is required to check the value of a parameter of Utterance ID. The system could fulfil different requirements according to its value (Figure 52).

```java
@Override
public void onUtteranceCompleted(String arg0) {
    Log.v("AndroidTextToSpeechActivity", arg0); 
    if(arg0.equals("keep on")){
        runOnUiThread(new Runnable() {
            
            @Override
            public void run() {
                String message; 
                if(_location)
                    message = _output + " keep on";
                else 
                    message = "keep on";
                
                _output = checkResponds(dialog_manager.getResponds(message));
                
                message_adaptor.notifyDataSetChanged();
                list_view.setSelection(list_view.getAdapter().getItemCount()-1);
                tts_Speak(_output, UtteranceID);

            }
        });
    } 
    else if(arg0.equals("Done")){
        if(this._currentState == 0){
            if(error_time >= 3){
                runOnUiThread(new Runnable() {
                    
                    @Override
                    public void run() {
                        dialogShow();
                        tts_Speak("Speech Recognition isn't really accurate. do you want to change to manual mode?","error");

                    }
                });
            }
        }
        else
            receiveVoiceInput();
    }
    else if(arg0.equals("End")){
        runOnUiThread(new Runnable() {
            
            @Override
            public void run() {
                Intent end_intent = new Intent(EIClinic.this, com.example.eiclini.EIEnd.class);
                startActivity(end_intent);

            }
        });
    }
}
```

**FIGURE 52 CHECKING UTTERANCE ID**
This figure indicates that there are 3 main utterance IDs, such as “keep_on”, “Done” and “End”. They could have different actions.

1) When the utterance ID is “keep_on”, system will automatically collect next responds from dialog manager.
2) Once it is “Done”, system could continue next turn of dialog especially when application use voice mode.
3) When the utterance ID is “End”, system will complete an emergency dialog and jump to an end activity, which will be introduced in the following section.

### 9.3.3 Location Tracker

According to system requirements analysed before, “EI Siri” is required to check emergency location in the beginning of application. Location could be shown when contacting Emergency Centre or receiving users’ requests. It was developed based on Google Location interface. I develop a particular class (named GPSTracker) to find latitude and longitude of a location via GPS on mobile devices. As shown in Figure 53, LocationManager is Instantiated to activate GPS and start location service. If it cannot be found, GPS setting will be called.

```java
private Location getLocation()
{
    try
    {
        this.location_manager = (LocationManager) m_context.getSystemService(Context.LOCATION_SERVICE);
        this.isEnableGPS = LocationManager.isProviderEnabled(LocationManager.GPS_PROVIDER); // checking GPS status
        this.isConnected = LocationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER); // checking Internet
        if((isEnableGPS && isConnected) || ((isEnableGPS || isConnected) && !this.isGPS))
        {
            if (isConnected) // if Internet is available
                Log.d("Network", "Connected");
            if(this.location_manager != null)
            {
                this.location = this.location_manager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                if(this.location != null)
                {
                    this.latitude = this.location.getLatitude(); // get latitude
                    this.longitude = this.location.getLongitude(); // get longitude
                }
            }
        }
        if(isGPS) // if get latitude and longitude
            if(this.location == null)
                this.location_manager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
                LocationManager.MANUAL_CONNECTION, 0, new LocationListener()
                {
                    if(this.location != null)
                    {
                        this.location = this.location_manager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                        if(this.location != null)
                        {
                            this.latitude = this.location.getLatitude(); // get Latitude
                            this.longitude = this.location.getLongitude(); // get Longitude
                        }
                    }
                }, null);
        } catch(Exception e) {
            return location;
        }
    }
}
```

**FIGURE 53** COLLECTING LATITUDE AND LONGITUDE OF A LOCATION

Latitude and longitude found from GPSTracker class will be transformed to a street address or other description of a location in the process of reverse geocoding via Geocoder. All
information relating to location (e.g. street address, city and postcode) are stored in a list and make a completed description in Figure 54.

```java
private String getLocation()
{
    String location_text = "";
    GPSTracker gps_tracker = new GPSTracker(this);
    if(gps_tracker.isGetLocation()){
        double latitude = gps_tracker.getLatitude(); // get latitude
        double longitude = gps_tracker.getLongitude(); // get longitude
        List<Address> address_list;
        Geocoder geo = new Geocoder(this, Locale.ENGLISH);
        try{
            address_list = geo.getFromLocation(latitude, longitude, 1);
            String address = address_list.get(0).getAddressLine(0); // get street address
            String city = address_list.get(0).getAdminArea(); // get city
            String code = address_list.get(0).getPostalCode(); // get post code
            if(address != null)
                location_text = address + ", ";
            if(city != null)
                location_text += city + ", ";
            if(code != null)
                location_text += code + ", ";
        }
        catch(IOException io){}
    }
    else{
        gps_tracker.alteringSettings();
    }
    return location_text;
}
```

**FIGURE 54 GETTING DESCRIPTION OF A LOCATION**

### 9.4 Core Implementation

#### 9.4.1 Dialog Manager

The dialog manager, as the core of “EI Siri”, is developed to process all voice XML files mentioned in the design chapter. In order to improve the efficiency of parsing xml files, I developed an interface named “DManger” to declare a set of essential functions (Figure 55).

```java
public interface DManager {
    public abstract String startUp(String _situation);
    public abstract String getResponds(String _input);
    public abstract String getNextStatement(String _node);
    public abstract String getNextNode(String _keyword);
    public abstract String getNextStep(String _nextStep);
    public abstract String getError();
}
```

**FIGURE 55 DManger INTERFACE**
As shown in figure above, there are several methods, including `startUp`, `getResponds`, `getStatement`, `getNextNode`, `gotoNextStep` as well as `getError`. I will explain all functions below.

1) The `startUp` method is designed to start different phases in an emergency dialog and to return the first respond.
2) The method of `getResponds` is the most important function in dialog manager. It will process text input from speech recognition and find corresponding answers immediately.
3) The `getStatement` method is designed to assist `getResponds` method to get respond of a specific node.
4) The method to `gotoNextStep` is used to search next node in dialog files, so that system could continue a new turn of dialog when a turn is finished.
5) The `getError` method is required to check the error statement when text input cannot be found in relevant XML files.

The whole dialog manage could be divided into 4 main phases, including general, situation analysis, level analysis and first aid. Each phase will overwrite all methods in “DManager” to fulfil their own goals. I will introduce them in the following sub sections.

“General” Phase

“General”, as the first phase, is developed to collect the emergency situation from users. In this phase, the system will process text input from speech recognition to find keywords relating to emergency situation. If keywords (such as emergency, ill and sick) cannot provide a clear situation, the system will transfer to the phase of situation analysis. Otherwise, the system is required to go to the corresponding phases, such as level analysis and first aid. There will be a parameter named “_currentStep” used to monitor current state of dialog manager. Once current state is not general, all text input will be directly sent to other phases according to state. This process is shown in an activity diagram in Figure 56.
This figure also shows that there will be an error statement provided to ask users to provide specific situation, when keywords in the input cannot be matched.

“Situation Analysis” Phase

This phase aims at diagnosing the emergency situation via a few questions provided by medical websites (e.g. NHS Choices [1] and British Red Cross [2]). The system is also required to find keyword (“yes” or “no”) in text input and provide a diagnosis according to diagnosis logic built in “situ_diag” XML file. The diagnosis logic was described in the previous chapter.

“Level Analysis” Phase

To be similar with the phase of “situation analysis”, users are only allowed to provide “yes” or “no” to answer questions from system. Level analysis is also developed to search keyword and check severity of problems depending on diagnosis logic in different sub-dialogs in “level_diag” XML file.
“First Aid” Phase

In the phase of “First aid”, I divided it into two possible states, as shown in Figure 57. When the level of problem is slight, the system should provide a first aid advice to users. When the problem is severe, the system must contact the Emergency Centre before providing rescue suggestions.

![Activity Diagram of Providing First Aid](image)

**FIGURE 57 ACTIVITY DIAGRAM OF PROVIDING FIRST AID**

This figure shows that users could willfully ask the system to repeat advice relating to first aid or finish the dialog directly at the end of this phase. If the dialog is completed, the method of gotoNextStep will be called to link the system to the general XML file to provide an ending statement.

On the other hand, in order to link the dialog manager with the prototype of “El Siri”, an object of dialog manager is created in EIClinic activity. And I developed a special method named “checkResponds” to modify the utterance ID of Text-to-Speech according to responses from the dialog manager (Figure 58).
private String checkResponds(String message) {
    String respond = message;
    if (message.contains("location")) {
        if (!_location.substring(0, 5).equals("Sorry"))
            respond = "Your location is " + _location;
        else
            respond = _location;
        this.UtteranceID = "keep_on";
        _isLocation = true;
    } else if (message.contains("&")) {
        respond = message.substring(0, message.length() -1);
        sendSMS();
        Toast.makeText(this, "Contacting Emergency Centre", Toast.LENGTH_SHORT).show();
    } else if (message.contains("*")) {
        respond = message.substring(0, message.length() -1);
        this.UtteranceID = "keep_on";
    } else if (message.contains("%")) {
        respond = message.substring(0, message.length() -1);
        this.UtteranceID = "Done";
        error_time++;
    } else {
        this.UtteranceID = "Done";
    }
    if (message.substring(message.length() - 5).equals("Siri.") )
        this.UtteranceID = "End";

    return respond;
}

**FIGURE 58 MODIFYING UTTERANCE ID FOR Responds**

### 9.4.2 Template Log

The function of “TempLog” is designed to record information (such as emergency and first aid advice) relating to emergency dialog in a template XML file. This XML file is created on SD card in the beginning of system. It contains a set of data nodes (Figure 59), described below:

1) **Date**: the date of emergency
2) **Start time**: when the emergency dialog is started
3) **Emergency**: the state of problem
4) **Level**: severity of problem
5) **First aid**: rescue advice
6) **Location**: emergency location
7) **Contact Time**: When contacting Emergency Centre
8) **IsContact**: whether system contact Emergency Centre
9) **End Time**: When dialog is Completed
10) **Num_Turn**: the number of turns in the dialog
I developed a class named "TempLog" with Singleton to change value of a particular node as well as read XML file. The method of "changeNodeValue" is developed to link system to template file and edit the value of specific node as shown in Figure 60.

```java
private static void changeNodeValue(String node_name, String value){
    try {
        InputStream _is = new BufferedInputStream(new FileInputStream(
            Environment.getExternalStorageDirectory() + "/El Siri/temp/temp.xml"));
        DocumentBuilderFactory xml_factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder xml_builder = xml_factory.newDocumentBuilder();
        Document xmlDoc = xml_builder.parse(_is);

        // Change the content of node
        Node nodes = xmlDoc.getElementsByTagName(node_name).item(0);
        nodes.setTextContent(value);

        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        DOMSource source = new DOMSource(xmlDoc);
        OutputStream os = new BufferedOutputStream(new FileOutputStream(
            Environment.getExternalStorageDirectory() + "/El Siri/temp/temp.xml"));
        StreamResult result = new StreamResult(os);
        transformer.transform(source, result);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

The "readXML" method is used to shown content of template file to end users, as shown in Figure 61.
public static String readXml(){
    String file_path = Environment.getExternalStorageDirectory() + "/EI Siri/temp/temp.xml";
    File file = new File(file_path);
    try {
        InputStream _is = new BufferedInputStream(new FileInputStream(file));
        DocumentBuilderFactory xml_factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder xml_builder = xml_factory.newDocumentBuilder();
        Document xmlDoc = xml_builder.parse(_is);

        Node emergency_node = xmlDoc.getElementsByTagName("emergency").item(0);
        String emergency = emergency_node.getTextContent();

        Node level_node = xmlDoc.getElementsByTagName("level").item(0);
        String level = level_node.getTextContent();

        Node location_node = xmlDoc.getElementsByTagName("location").item(0);
        String location = location_node.getTextContent();

        Node first_aid_node = xmlDoc.getElementsByTagName("first_aid").item(0);
        String first_aid = first_aid_node.getTextContent();

        Node start_time_node = xmlDoc.getElementsByTagName("start_time").item(0);
        String start_time = start_time_node.getTextContent();

        Node contact_time_node = xmlDoc.getElementsByTagName("contact_time").item(0);
        String contact_time = contact_time_node.getTextContent();

        Node end_time_node = xmlDoc.getElementsByTagName("end_time").item(0);
        String end_time = end_time_node.getTextContent();

        Node date_node = xmlDoc.getElementsByTagName("date").item(0);
        String date = date_node.getTextContent();

        Node num_turn_node = xmlDoc.getElementsByTagName("num_turn").item(0);
        String num_turn = num_turn_node.getTextContent();

        String Message = "Date : " + date + "\n";
        Message += "Start Time : " + start_time + "\n";
        Message += "Emergency Situation : " + emergency + "\n";
        Message += "Level Of Illness : " + level + "\n";
        Message += "First Aid : " + first_aid + "\n";
        Message += "Location : " + location + "\n";
        Message += "Contact Time : " + contact_time + "\n";
        Message += "End Time : " + end_time + "\n";
        Message += "The Number of Turns : " + num_turn + "\n";

        return Message;
    } catch (ParserConfigurationException e) {
        e.printStackTrace();
        return null;
    } catch (SAXException e) {
        e.printStackTrace();
        return null;
    } catch (IOException e) {
        e.printStackTrace();
        return null;
    }
}
9.5 GUI IMPLEMENTATION

“EI Siri” contains 4 main screens (so-called activity in Android applications), including Splash, EIMainActivity, EIClinic and EIEnd. This section will introduce layout XML file and corresponding code of each screen.

9.5.1 SPLASH

The Splash activity is used to show the logo of application and make a preparation in the beginning of “EI Siri”. It is developed with a set of actions (e.g. checking SD card, creating system folder and template file, as well as get emergency location). In the development of Splash activity, there is a parameter named “SPLASH_DISPLAY_LENGTH”, which defines the delay time. System cannot jump to next activity (so-called EIMainActivity) until the delay time is reached, as shown in Figure 62.

```java
@Override
public void run() {
    _location = getLocation();
    if (hasSdcard()){
        createDirInfo("/EI Siri/temp");
        createTempXML();
        Intent mainIntent = new Intent(Splash.this,EIMainActivity.class);
        mainIntent.putExtra("location", _location);
        Splash.this.startActivity(mainIntent);
        Splash.this.finish();
    } else
        dialogShow();
}
}, SPLASH_DISPLAY_LENGTH);
```

FIGURE 62 JUMPING TO NEXT ACTIVITY

9.5.2 EI MAIN ACTIVITY

In this main activity, the system needs to show a disclaimer about “All information and activity for emergency rescue are for medical suggestion only. Users are encouraged to confirm information with other sources or contact with Emergency Centre (999) directly”. I developed a method of “isConnected” checking current Internet State (Figure 63).

```java
private boolean isConnected(){
    ConnectivityManager connect_manager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    if (connect_manager.getActiveNetworkInfo() != null && connect_manager.getActiveNetworkInfo().isAvailable() && connect_manager.getActiveNetworkInfo().isConnected())
        return true;
    else
        return false;
}
```

FIGURE 63 CHECKING INTERNET STATUS
EI main activity creates an image button of “start” in layout XML file, as shown in Figure 64(a). The system checks Internet status using the method of “isConnected” when this button is pressed. Once the Internet is found, application could jump into EIClinic with currentState of “0”, otherwise the value of currentState is “1”, as shown in Figure 64(b). “0” means voice input mode, while “1” means manual mode in system.

![Image Button](image1.png)

(a) “Start” Button in Layout XML

```java
if(isConnected()){
    Intent intent = new Intent(this, com.example.eisiri.EIClinic.class);
    intent.putExtra("location", this._location);
    intent.putExtra("currentState", "0");
    startActivity(intent);
} else
    dialogsShow();
```

(b) Method of “Start” Button

**FIGURE 64 EIMAINACTIVITY BUTTON ACTION**

9.5.3 EI CLINIC

This activity focuses on building a natural dialog for emergency. It consists of two main input modes: voice mode and manual mode. In voice mode, there are two image buttons, one is “voice” button and another one is “manual” button. They are declared in a layout XML file, as shown in Figure 65.

![Image Button](image2.png)

(a) “voice” Button
I developed "voice" button to control the speech recognition to collect speech data from users. The purpose of "manual" button is to change activity to manual input mode. When this button is pressed, there may be a dialog shown to change input mode (Figure 66). And when the number of errors is more than 3, this dialog is also shown.

```java
protected void dialogShow(){
    AlertDialog.Builder dialog_builder = new Builder(UIClinic.this);
    dialog_builder.setMessage("Do you want to change to Manual Mode?");
    dialog_builder.setTitle("Information");
    dialog_builder.setPositiveButton("Okay", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int arg1) {
            current_state = 1;
            vf.showNext();
        }
    });
    dialog_builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            error_time = 0;
            dialog.dismiss();
        }
    });
    dialog_builder.create().show();
}
```

**FIGURE 66 SHOWING DIALOG TO CHANGE INPUT MODE**

In manual mode, there is only one button named "send", which is used to send text input to the mobile application. The text input from users will be processed (e.g. removing capital letters and punctuations) before sending it to dialog manager, as shown in Figure 67.

```java
String user_text = user_editText.getText().toString();
_input = user_text.toLowerCase().replaceAll("[\?!,.;']", "");
```

**FIGURE 67 PROCESSING TEXT INPUT FROM USERS**

On the other hand, EI Clinic activity is required to display all inputs and outputs on screen. I designed a special layout XML file for the message box, in which there are several attributes (such as sender, message and sending-time) in Figure 68.
In order to list all messages, I develop two essential classes. The first one is named `NewMessageItem` which focuses on storing all message information as a unit. System could create an object for each message. Another class is `MessageAdapter` based on `BaseAdapter`. It creates an array list to store `NewMessageItem` objects and list each list item in a list view on screen (Figure 69).

```java
@Override
public View onCreateViewHolder(ViewGroup parent, int viewType) {
    ViewHolder viewHolder = null;
    if (viewHolder == null) {
        viewHolder = new ViewHolder();
    }
    viewHolder.senderView = (TextView) itemView.findViewById(R.id.sender);
    viewHolder.messageView = (TextView) itemView.findViewById(R.id.message);
    viewHolder.timeView = (TextView) itemView.findViewById(R.id.time);
    viewHolder.setTag(viewHolder);
    return viewHolder;
}

@Override
public void onBindViewHolder(ViewHolder viewHolder, int position) {
    viewHolder.senderView.setText(arr.get(position).getSender());
    viewHolder.messageView.setText(arr.get(position).getMessage());
    viewHolder.timeView.setText(arr.get(position).getTime());
}
```

**FIGURE 68 LIST VIEW ITEM IN LAYOUT XML FILE**

**FIGURE 69 SETTING LIST VIEW ITEMS**
Hence, once there is a new message provided by end user or system, list view I designed will be updated and required to focus on the final item. Its code is shown in Figure 70.

```java
message_adapter.add(new NewMessageItem("User", _input, getTime()));
message_adapter.notifyDataSetChanged();
list_view.setSelection(list_view.getAdapter().getCount()-1);
```

**FIGURE 70 UPDATING LIST VIEW AND SETTING FOCUS**

9.5.4 EI END

In the final screen, there are two main components, one is text view and another one is image button. They are declared in layout XML file of “activity_end” (Figure 71).

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

Text view is used to display all information relating to emergency dialog using the “readXML” method in TempLog class. The system could remind users to check relevant information via Text-To Speech interface, as shown in Figure 72.
FIGURE 72 SHOWING EMERGENCY INFORMATION

Moreover, system provides a “restart” button to allow users to start application again. Once this button is activated, application will jump to the first activity – Splash. Its code is shown in Figure 73.

FIGURE 73 RESTARTING “EI SIRI”
SYSTEM TESTING AND EVALUATION

10.1 OVERVIEW

This chapter will discuss and demonstrate testing tasks based on all requirements (as mentioned in chapter 5) as well as evaluation analysis. I tested my application on a mobile device and listed all results to confirm that all bugs are resolved in this stage. All testing results are described in the next section. Moreover, in order to testify the hypothesis in this project, I designed an experiment with a questionnaire to compare “El Siri” I proposed with other existing applications (e.g. the “Esagil.org” website [36] and the “British Red Cross” application [31]). I will analyze its results in the section on evaluation.

10.2 TESTING DESCRIPTION

There are a significant number of tasks I tested before in Table 7.

<table>
<thead>
<tr>
<th>#</th>
<th>Tasks</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Check SD Card</td>
<td>If SD card cannot be found, a dialog is shown and system will be closed.</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>Check Internet</td>
<td>If Internet cannot be found, a dialog is shown to display errors and change input to manual mode.</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>Press “Voice” Button</td>
<td>The intent of Speech Recognition is built.</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>Press “Manual” Button</td>
<td>A dialog is shown to change application to manual mode.</td>
<td>✓</td>
</tr>
<tr>
<td>5</td>
<td>The number of error is more than 3 times</td>
<td>A dialog is shown to change application to manual mode.</td>
<td>✓</td>
</tr>
<tr>
<td>6</td>
<td>Check keyword in text input</td>
<td>If keyword in input cannot be matched, an error statement will be displayed and converted to speech.</td>
<td>✓</td>
</tr>
<tr>
<td>7</td>
<td>Severity of problem is high</td>
<td>System needs to send a SMS to Emergency Centre directly.</td>
<td>✓</td>
</tr>
<tr>
<td>8</td>
<td>Request to check location</td>
<td>A description of emergency location is displayed and converted to speech.</td>
<td>✓</td>
</tr>
<tr>
<td>9</td>
<td>Press “Send” button</td>
<td>All punctuations will be removed and capital letters are changed into lowercase letters.</td>
<td>✓</td>
</tr>
<tr>
<td>10</td>
<td>Dialog is completed</td>
<td>All information relating to emergency will be displayed on screen. System could notice user in speech.</td>
<td>✓</td>
</tr>
<tr>
<td>11</td>
<td>Press “Restart” button</td>
<td>A dialog is shown to restart application.</td>
<td>✓</td>
</tr>
<tr>
<td>12</td>
<td>Record Emergency Information</td>
<td>Emergency information will be recorded into template XML file separately.</td>
<td>✓</td>
</tr>
</tbody>
</table>

*TABLE 7 TESTING DESCRIPTIONS*
10.3 Evaluation

10.3.1 Participant Background

In the evaluation phase, 20 participants tested my application compared to existing websites, and filled the questionnaire. In order to lead a clear analysis, all participants are separated according to several conditions or backgrounds, such as gender, mobile application familiarity, voice interface familiarity as well as language background. These conditions may affect the result analysis of this experiment. There are several figures presenting the number of the participants with different conditions in Figure 74.

![Charts showing gender, language, and mobile application familiarity](image)

(a) Gender  
(b) Language  
(c) Familiarity of Mobile Application and Voice Interface

**FIGURE 74 PERCENTAGE OF PARTICIPANTS WITH DIFFERENT BACKGROUNDS**

As we see from the first chart in Figure 74, the male accounts for 65% of the total participants, higher than that of the female (35%). The second pie chart shows that there are totally 12 non-native speakers, which is more than the sum of the number of native speakers and Europeans (8).
In addition, in terms of professional background, there are less than half of the participants (9) being familiar with mobile apps and voice interfaces. The majority of the rest of the testers (6 people) are only familiar with mobile apps. The number is nearly twice as much as that of people who can know about neither mobile apps nor voice interfaces (only 3).

In the next sub-section, I will analyse the answers from all participants respectively according to these conditions.

### 10.3.2 Result analysis

In this experiment, the final result analysis is made up of three essential parts, including activity efficiency, information satisfaction as well as time cost. Both activity efficiency and information satisfaction are combined to rate the success ($k$) mentioned in the PARADISE evaluation framework and time cost is used to represent dialog cost ($C$) \cite{35}. In order to consider different aspects, I separate all case cards into two parts, one for clear situations and another one for unclear situations. Participants are required to test case cards with clear and unclear situations and fill related questionnaire respectively. The questionnaire contains 4 main questions, as listed below:

a) Is the system easy to use for diagnosing emergency situation?

b) Is the system easy to use for checking severity of the problem?

c) Is the system easy to use for providing first aid advice?

d) Did this system provide you with all information that you need?

I will describe these questions using “a, b, c, and d”. Question a, b and c are analysed for rating activity efficiency of the “EI Siri” and other applications (e.g. the “Esagil.org” website \cite{36} and the “British Red Cross” application \cite{31}). For question a, b and c, they are scaled using 6 points. The more the point is, the higher the degree of satisfaction is.

In the following subsections, I describe and compare activity efficiency, information satisfaction and time cost in clear situations and unclear situations.

#### Activity Efficiency

All participants are required to fill question a, b and c after each case card. These questions could show whether the system could be efficient to reach corresponding strategies, including diagnosing situation and severity, as well as providing first aid advice. This subsection could compare the answers of each question in the clear and unclear situations using the “EI Siri” and the combination of the “Esagil.org” website \cite{36} and the “British Red Cross” application \cite{31}. I will describe these questions one by one in the following.
Question A considers whether the system is easy to diagnose emergency situation. There is a form presenting all answers with different situations and applications in Table 8.

<table>
<thead>
<tr>
<th>Situation</th>
<th>Application</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unclear</td>
<td>El Siri</td>
<td>2.5%</td>
<td>2.5%</td>
<td>5%</td>
<td>12.5%</td>
<td>37.5%</td>
<td>40%</td>
<td>4.98 / 83%</td>
</tr>
<tr>
<td>Websites</td>
<td></td>
<td>20%</td>
<td>20%</td>
<td>15%</td>
<td>30%</td>
<td>12.5%</td>
<td>2.5%</td>
<td>3.03 / 50.5%</td>
</tr>
</tbody>
</table>

**TABLE 8 COLLECTION OF ANSWERS FOR QUESTION A**

As shown in the table above, both the “El Siri” and the combination of the “Esagil.org” website [36] and the “British Red Cross” application [31] were not applicable for clear situations, because the situation diagnosis is not necessary. In the unclear situation, nearly 37.5% and 40% of participants could agree and strongly agree respectively that the “El Siri” is easy to use to diagnose emergency situations, which are significantly higher than the percentage of subjects who supports the combination of the “Esagil.org” website [36] and the “British Red Cross” application [31] (12.5% and 2.5% respectively). Half of the testers were not be satisfied with the performance of the combination of the “Esagil.org” website [36] and the “British Red Cross” application [31]. At the same time, the average answer of question A for the “El Siri” (about 4.98) is more than that for other applications, as shown in Figure 75.

**FIGURE 75 ANALYSIS OF ANSWERS FOR QUESTION A**

Within subject, I require participants to rate the Question A in score and compared two paired groups of answers. Hence, I did a Wilcoxon test to measure the significance between the “El Siri” and the combination of the “Esagil.org” website [36] and the “British Red Cross” application [31] in SPSS [38]. Due to the face that the functionality of diagnosing situation in clear situations was not applicable, I just considered the significance of Question A in unclear situations. The result shows that the “El Siri” helps users to diagnose emergency...
situation significantly easier than the combination of the “Esagil.org” website [36] and the “British Red Cross” [31] (p = 0.000).

Question B aims at rating the efficiency of systems in terms of checking problem severity. There is a form presenting all answers with different situations and applications in Table 9.

<table>
<thead>
<tr>
<th>Situation</th>
<th>Application</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>EI Siri</td>
<td>2.5%</td>
<td>2.5%</td>
<td>2.5%</td>
<td>10%</td>
<td>32.5%</td>
<td>45%</td>
<td>5.13 / 85.5%</td>
</tr>
<tr>
<td></td>
<td>Websites</td>
<td>42.5%</td>
<td>17.5%</td>
<td>7.5%</td>
<td>17.5%</td>
<td>15%</td>
<td>5%</td>
<td>2.45 / 40.83%</td>
</tr>
<tr>
<td>Unclear</td>
<td>EI Siri</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>15%</td>
<td>32.5%</td>
<td>52.5%</td>
<td>5.38 / 89.67%</td>
</tr>
<tr>
<td></td>
<td>Websites</td>
<td>50%</td>
<td>17.5%</td>
<td>5%</td>
<td>25%</td>
<td>2.5%</td>
<td>0%</td>
<td>2.13 / 35.5%</td>
</tr>
</tbody>
</table>

**TABLE 9 COLLECTION OF ANSWERS FOR QUESTION B**

In terms of the clear situation, this table shows that there are 77.5% of the testers providing a positive answer for the “EI Siri” in the question B, which is about 3 times as high as that of testers who support the performance of the combination of the “Esagil.org” website [36] and the “British Red Cross” application [31] (only 20%).

Similarly, in terms of the unclear situation, people supporting “EI Siri” takes up 85% in total, while the majority of testers (72.5%) do not approve of other applications (e.g. the “Esagil.org” website [36] and the “British Red Cross” application [31]).

Totally, the average answers of the “EI Siri” in the clear situation and the unclear one are 5.13 and 5.38, which is approximately twice as much as that of other applications (2.45 in clear situations and 2.13 in unclear situations), as shown in Figure 76.
To be similar with Question A, I did a Wilcoxon test to measure the significance of Question B in SPSS [38]. As a result, the “EI Siri” could check severity of the problem significant easier than the combination of the “Esagil.org” website [36] and the “British Red Cross” application [31], no matter in clear situations (p = 0.000) or unclear situations (p = 0.000).

The purpose of Question C is to confirm which system is easy to provide first aid advice to patient. There is a form presenting all answers with different situations and applications in Table 9.

**Question C: Is the system easy to use for providing first aid advice?**

<table>
<thead>
<tr>
<th>Situation</th>
<th>Application</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>EI Siri</td>
<td>5%</td>
<td>5%</td>
<td>0%</td>
<td>2.5%</td>
<td>35%</td>
<td>52.5%</td>
<td>5.15 / 85.83%</td>
</tr>
<tr>
<td></td>
<td>Websites</td>
<td>2.5%</td>
<td>7.5%</td>
<td>15%</td>
<td>20%</td>
<td>35%</td>
<td>20%</td>
<td>4.38 / 73%</td>
</tr>
<tr>
<td>Unclear</td>
<td>EI Siri</td>
<td>0%</td>
<td>2.5%</td>
<td>0%</td>
<td>7.5%</td>
<td>47.5%</td>
<td>42.5%</td>
<td>5.28 / 88%</td>
</tr>
<tr>
<td></td>
<td>Websites</td>
<td>12.5%</td>
<td>7.5%</td>
<td>7.5%</td>
<td>22.5%</td>
<td>25%</td>
<td>25%</td>
<td>4.15 / 69.17%</td>
</tr>
</tbody>
</table>

*TABLE 10 COLLECTION OF ANSWERS FOR QUESTION C*

This table indicates that more than half of participants are satisfied with performance of both the “EI Siri” and the combination of the “Esagil.org” website [36] and the “British Red Cross” application [31] in no matter clear situations or unclear situation. It means that they could easily provide first aid advice.

However, although it is not difficult for both of the systems to rescue the patient, the “EI Siri” is better overall than the combination of “Esagil.org” website [36] and “British Red Cross” application [31]. The average answer for the “EI Siri” (about 5.15 in clear situations and 5.275 in unclear situations) are more than that for other applications (nearly 4.38 in clear situations and 4.15 in unclear situations), as shown in Figure 78.

*FIGURE 77 ANALYSIS OF ANSWERS FOR QUESTION C*
Through Wilcoxon testing, in clear situations, El Siri could provide first aid advice significantly easier than the combination of the “Esagil.org” website [36] and the “British Red Cross” application [31] (p = 0.007). In unclear situation, El Siri also provides first aid advice significantly easier than other applications (p = 0.000).

Information Satisfaction

On the other hand, the 4th question considers whether the system could provide enough information relating to the emergency and corresponding first aid advice. Hence, I will also analyse all data from the participants without considering emergency situation, whether clear situation or unclear one. There are two bar charts presenting the final results from the combination of the “Esagil.org” website [36] and the “British Red Cross” application [31] as well as the “El Siri” I proposed respectively in Figure 78.

As shown in this figure, the majority of the participants (nearly 79%) thought that the “El Siri” could provide all information they need in a clear emergency situation, which is significantly more than that of testers who support the combination of the “Esagil.org” website [36] and the “British Red Cross” application [31] (nearly 42%). 49% of the testers in this experiment thought that the website of “Esagil.org” [36] and the application of “British Red Cross” [31] might partially provide information relating to emergency situation and first aid advice.

On the other hand, there are only 4% and 9% of the testers cannot be satisfied with the “El Siri” and other applications respectively.

In order to measure the significance, I did a McNemar’s test for two different applications using SPSS [38]. As shown in the result, the “El Siri” could show more information than the combination of the “Esagil.org” website [36] and the “British Red Cross” application [31] (p = 0.000).
**Time Cost**

As mentioned in the chapter of methodology, the evaluation analysis contains two main parts, one for success (k) and another one for dialog cost (Ci) [35]. In this experiment, time cost could be viewed as the dialog cost used for rating different emergency application. There is a chart showing the time cost of the “El Siri” and the combination of the “Esagil.org” website [36] and the “British Red Cross” application [31] in clear and unclear situations in Figure 79.

![Average Time Cost Chart](image)

**FIGURE 79 TIME COST USING DIFFERENT APPLICATIONS**

For the cost time, the paired samples t-test [38] is the most suitable way to measure the significance between the “El Siri” and the combination of the “Esagil.org” website [36] and the “British Red Cross” [31]. In terms of clear situations, the “El Siri” (about 1’57”) spent twice as much time as that spent by other websites (nearly 1’05”). As shown in the result of the t-test, the combination of the “Esagil.org” website [36] and the “British Red Cross” [31] spend significantly less time on emergency help than the “El Siri” (p = 0.000).

On the other hand, there is not difference in the cost time between the “El Siri” (about 2’58”) and the combination of the “Esagil.org” website [36] and the “British Red Cross” [31] (nearly 3’19”) in unclear situations (p = 0.155).

**10.3.3 Evaluation Discussion**

With regard to the result of evaluation, the hypothesis about “Using proposed mobile emergency application is faster and better to provide aid than using existing web-based first-aid websites” mentioned before is partially proven. The “El Siri” I proposed in general fulfils its objectives in terms of diagnosis and first aid. This application is better but not faster to perform relevant activities (such as diagnosing situations, checking severity, and
providing first aid). Although the average speed of “El Siri” is lower than the existing websites, the cost time is also less than 4 minutes. It could also provide enough information for specific emergency situation.

In general, the majority of users prefer to apply the “El Siri” for an emergency. This system could not be limited by the Internet, due to the fact that system could automatically change to manual mode once the Internet cannot be found. By contrast, they are annoyed that website (such as “Esagil.org” [36]) does not work without network connection.

Moreover, the “El Siri” provides a functionality of checking severity of the problem automatically, which cannot be found in the combination of the “Esagil.org” website [36] and the “British Red Cross” application [31]. Instead, users are required to search a lot of relevant information online to confirm whether they need to go to hospital.

According to feedback from non-native speakers, the “El Siri” is better for providing first aid advice than another one, because it is really difficult for them to directly get helpful information from other applications (e.g. the “Esagil.org” website [36] and the “British Red Cross” application [31]). For example, non-native speakers will read whole article and check the meaning of new words to find useful information, because of the limitation of English skills.

However, some of the participants were not be satisfied with the functionalities of “El Siri”. They provide a set of negative points for “El Siri”. For instance, the “El Siri” could allow users to talk with it in a natural dialog but did not include all potential vocabulary related to each emergency situation. This problem was observed when attempting to ask a help for unclear situation in the beginning of dialog. Users with different language backgrounds will start an emergency dialog with different words or sentences. Some of the words may not be included in this system, so that users must remember which words they could use in the beginning of dialog. It cannot be found in the website of “Esagil.org” [36], but users must spend too much time on choosing possible symptoms on the website rather than helping patient.

On the other hand, another issue in the “El Siri” is that system did not allow users to make mistakes in communication. It means there is not any chance to repair their answers once relevant question is completed. This could be improved in the future version.

## 10.3.4 Improvement Based On Evaluation

Though evaluation discussion above, I collect several problems relating to performance of the “El Siri”. In order to improve its efficiency, I discuss some of the problems, including enlarging the vocabulary and improving first aid advice. I will describe them below.
Enlargement of Vocabulary

There are a significant number of choices to start an emergency situations (e.g. emergency keywords as well as specific symptoms as well) for users with different language backgrounds. In order to deal with this issue, I have modified the voice-based xml file and added more potential keywords into situation list. Hence, system could go to next phase according to relevant keywords. For example, system will start up the functionality of diagnosing emergency situation once a specific symptom (e.g. chest pain and anxiety) is inputted in the beginning of dialog, as shown in Figure 80.

```xml
<answer value="emergency" next="situation"/>
<answer value="problem" next="situation"/>
<answer value="accident" next="situation"/>
<answer value="pain" next="situation"/>
<answer value="sick" next="situation"/>
<answer value="ill" next="situation"/>
<answer value="trouble" next="situation"/>
<answer value="pain" next="situation"/>
<answer value="tiredness" next="situation"/>
<answer value="anxiety" next="situation"/>
<answer value="lightheaded" next="situation"/>
<answer value="weak" next="situation"/>
<answer value="wheezing," next="situation"/>
<answer value="vomiting" next="situation"/>
<answer value="diarrhoea" next="situation"/>
<answer value="cramps" next="situation"/>
<answer value="difficulty" next="situation"/>
<answer value="difficult" next="situation"/>
<answer value="burned" next="situation_bs"/>
<answer value="burnard" next="situation_bs"/>
<answer value="burnu" next="situation_bs"/>
<answer value="burnt" next="situation_bs"/>
<answer value="scalded" next="situation_bs"/>
<answer value="sprains" next="situation_ss"/>
<answer value="sprain" next="situation_ss"/>
<answer value="sprained" next="situation_ss"/>
<answer value="broken" next="situation_ss"/>
<answer value="strains" next="situation_ss"/>
<answer value="food poisoning" next="situation_fp"/>
<answer value="asthma attack" next="situation_oa"/>
<answer value="heart attack" next="Unclear"/>
```

**FIGURE 80 EXAMPLE OF ENLARGEMENT OF VOCABULARY**

Improvement of First Aid Advice

In order to improve the effect of first aid advice, I searched for more information relating to how to provide rescue measures and list its compact edition in first-aid xml file. It could be meaningful to improve practicality of rescue measure especially when the situation is very urgent.
SUMMARY

11.1 OVERVIEW

This chapter will discuss the achievement and limitation of this project. Moreover, I also discuss its evaluation and future work.

11.2 ACHIEVEMENT & LIMITATION

11.2.1 ACHIEVEMENTS

As the result of this project, the hypothesis about “Using proposed mobile emergency application is faster and better to provide aid than using existing web-based first-aid websites” could be proved significantly. The “Emergency Intelligent Siri” I proposed in the project could fulfil all of core requirements. In other words, this project developed an emergency voice-based interface which could meets system requirements and user requirements listed in the following:

✓ As the system is built with a set of voice-based XML files based on a wide range of potential keywords, it enables users to communicate with the system in a natural dialog, but not to remember the specific commands.
✓ The techniques used in the development of this system allow it to perform the whole process (including diagnosing situations, checking severity and providing first aid advice) with only one screen in a short time.
✓ The system in this project eliminates the effect of the Internet, owing to the fact that it is required to be installed and performed independently on mobile devices. Although the Google Voice API is supported by Google server, this system allows users to enter what they want to know without the Internet Connection.
✓ It uses XML Parse to record and read template data (so-named final result) in a template file, so that users could check details after emergency dialog.
✓ The “EI Siri” performs all process and show an accurate result in each stage, such as situations, severity of the problem and specific first aid advice.

11.2.2 LIMITATIONS

The project also has a set of aspects which should be considered in the future version. These aspects bring several limitations in this project. Firstly, as VoiceXML interpreter cannot be independently implemented on android platform, application I proposed is lack of more complex dialog logic. It means some important functionality, such as repair, is difficultly
implemented in this system. And the majority of time was spent on considering how to create a set of VoiceXML-simulated files and how to build a particular interpreter for them. The time spent on studying new voice-based XML theory and performing a significant number of coding affected this project negatively.

Secondly, although the system could contact Emergency Centre, it cannot be used in practice, because there is not a special application to receive all emergency texts and arrange the ambulance for them. Hence, I just developed the “El Siri” send a SMS to the specific mobile number to simulate the contacting process in the real world.

These issues mentioned before will be considered in the future work in the next section. I will discuss their solutions in details.

11.3 Future Work

In the terms of future work of this project, some features or issues described in the limitations could be implemented, such as repair mechanism in the dialog, voice-simulated XML implementation on Android platform and Emergency Centre responder.

The first aspect of future work is to implement a repair mechanism in the “El Siri”. During the diagnosis phase in the system, users may possibly make some mistakes when answering questions, because of tension or not familiar with the situation. This mechanism could allow users to repair the answer of a specific question whenever they want. In the other hand, users could return to any a question and answer it again. On the other hand, users do not need to answer other questions which answers are correct.

Another aspect in the future work focuses on how to improve the efficiency of the dialog manager, which is the core of application. This project built the “El Siri” with a set of voice XML files, which is created based on the syntax and grammar of VoiceXML file. These XML files are processed on Android mobile devices instead of VoiceXML. However, due to lack of the support of the grammar logic in VoiceXML, the voice-simulated XML files cannot fulfil more complex requirements in a natural dialog. In the future work, a more powerful voice-based XML file needs to be created to deal with the more complex dialog relations. This is also the foundation of building repair mechanism in the “El Siri”.

Finally, in order to improve the efficiency of emergency dialog agent, an Emergency responder could be developed for Emergency Centre (999). This responder is used to link the “El Siri” with Emergency Centre. It aims at receiving all emergency messages and arranging the ambulances for them after processing these messages. This application need to toward relevant emergency information (e.g. the situation, basic symptoms and location).
to the ambulance which is closer to the emergency location. It could shorten the rescue time and improve the rescue efficiency.

### 11.4 Conclusion

This project is proposed to develop an emergency voice-based interface focusing on helping the patient in an emergency to diagnose situations, to check its severity as well as provide useful first aid advice via a spoken dialog agent. In particular, it reviewed the relevant literature and summarized the core technology in Natural Language Processing and the main architecture of spoken dialog agent. The project mainly focuses on building an independent spoken dialog system integrated with the existing speech input interface (e.g. Google Speech API) and speech output interface (e.g. Google Text-To-speech API).

As the core part of this project, an emergency voice agent (so-called “EI Siri”) was developed on Android platform. It is important to mention that during the development phase, application is required to provide clear logical relations between different situations and different symptoms, which will positively affect the efficiency of the system in terms of the situation and severity diagnosis. The system aims to use voice-based XML files to build a useful dialog tree. The dialog tree allows users to find corresponding information when inputting complete sentences or keywords in the form of speech.

Finally, in order to verify the effectiveness and practicability of the system, this project provided a contrastive experiment on the topic of which system could easily help the patient in an emergency. Through significance testing, “EI Siri” I proposed in this project is significantly easier to provide help for emergency situation than other existing websites. However, In terms of the speed of application, existing websites deal with problem significantly faster than “EI Siri” in the clear situations, but there is no significant difference between websites and “EI Siri” in the unclear situations.

During the evaluation phase, more of the comments and findings from testers have been implemented in the final system. Despite this, there is a lot of meaningful information which could be considered in the future version.
BIBLIOGRAPHY


**APPENDIX A: QUESTIONNAIRES FOR THE “EI SIRI”**

### Evaluation Questionnaire for Intelligent Emergency Assistant

<table>
<thead>
<tr>
<th>Name: ____________________________</th>
<th>Gender: [ ] Male / [ ] Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age: _____________________________</td>
<td></td>
</tr>
<tr>
<td>Familiar with mobile app: [ ] Yes / [ ] No</td>
<td>Familiar with voice interface: [ ] Yes / [ ] No</td>
</tr>
<tr>
<td>Native speaker: [ ] Yes / [ ] No</td>
<td>European: [ ] Yes / [ ] No</td>
</tr>
<tr>
<td>Date/Time: ___________________________</td>
<td></td>
</tr>
</tbody>
</table>

**Note:**
1. Each tester could have 4 different case cards, which contain symptoms of situation individually.
2. Please read each case card to diagnose emergency and to find first aid advice.
3. Please fill questionnaire for each case card.

"I give consent for this data to be used for research purposes. I understand that I can withdraw from this experiment at any time."

Signed: ___________________________  Date: ________ / ________ / ________
### Case Card

**Patient:** A child in 10 years old  
**Situation:** Burnt  
**Symptoms:** His arm is burnt when he joins a BBQ party with his parents There are some clothes stuck to the burn.

<table>
<thead>
<tr>
<th>Start Time:</th>
<th>End Time:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Which System do you use for evaluation?  
☐ ESagil.org + British Red Cross  
☐ EI Siri

The System is easy to use for:

<table>
<thead>
<tr>
<th>Activity</th>
<th>Strongly Disagree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagnosing Emergency Situation</td>
<td>[1,2,3,4,5,6]</td>
<td></td>
</tr>
<tr>
<td>Checking Severity of The Problem</td>
<td>[1,2,3,4,5,6]</td>
<td></td>
</tr>
<tr>
<td>Providing First Aid Advice</td>
<td>[1,2,3,4,5,6]</td>
<td></td>
</tr>
</tbody>
</table>

Did this system provide you with all information that you need?  
[YES, NO, PARTIALLY]

---

### Case Card

**Patient:** A pregnant woman  
**Situation:** Unclear Situation  
**Symptoms:** She could breathe normally. But she has diarrhoea and frequently vomits with a slight stomach cramps. She has a high fever.

<table>
<thead>
<tr>
<th>Start Time:</th>
<th>End Time:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Which System do you use for evaluation?  
☐ ESagil.org + British Red Cross  
☐ EI Siri

The System is easy to use for:

<table>
<thead>
<tr>
<th>Activity</th>
<th>Strongly Disagree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagnosing Emergency Situation</td>
<td>[1,2,3,4,5,6]</td>
<td></td>
</tr>
<tr>
<td>Checking Severity of The Problem</td>
<td>[1,2,3,4,5,6]</td>
<td></td>
</tr>
<tr>
<td>Providing First Aid Advice</td>
<td>[1,2,3,4,5,6]</td>
<td></td>
</tr>
</tbody>
</table>

Did this system provide you with all information that you need?  
[YES, NO, PARTIALLY]
## Case Card

### No. 3

**Patient:** A student in Heriot-Watt University  
**Situation:** Unclear Situation  
**Symptoms:** He has difficulty breathing and chest pain. He feels anxiety, but not lightheaded. He has a slight wheezing when he breathes. He is too breathless to speak.

**Start Time:** ________________  
**End Time:** ________________

**Which System do you use for evaluation?**  
☐ ESagil.org + British Red Cross  
☐ EI Siri

### The System is easy to use for:

<table>
<thead>
<tr>
<th></th>
<th>Strongly Disagree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Diagnosing Emergency Situation</strong></td>
<td>1 2 3 4 5 6</td>
<td></td>
</tr>
<tr>
<td><strong>Checking Severity of The Problem</strong></td>
<td>1 2 3 4 5 6</td>
<td></td>
</tr>
<tr>
<td><strong>Providing First Aid Advice</strong></td>
<td>1 2 3 4 5 6</td>
<td></td>
</tr>
</tbody>
</table>

**Did this system provide you with all information that you need?**  
Yes  | No  | Partially

### No. 4

**Patient:** User  
**Situation:** Your leg is broken.  
**Symptoms:** You don’t have any swelling around a joint and lumps, but you cannot move it.

**Start Time:** ________________  
**End Time:** ________________

**Which System do you use for evaluation?**  
☐ ESagil.org + British Red Cross  
☐ EI Siri

### The System is easy to use for:

<table>
<thead>
<tr>
<th></th>
<th>Strongly Disagree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Diagnosing Emergency Situation</strong></td>
<td>1 2 3 4 5 6</td>
<td></td>
</tr>
<tr>
<td><strong>Checking Severity of The Problem</strong></td>
<td>1 2 3 4 5 6</td>
<td></td>
</tr>
<tr>
<td><strong>Providing First Aid Advice</strong></td>
<td>1 2 3 4 5 6</td>
<td></td>
</tr>
</tbody>
</table>

**Did this system provide you with all information that you need?**  
Yes  | No  | Partially

**Comments on your rating for this system:**

**Signed :** ____________________  
**Date :** _____/_____/_______

[96]
APPENDIX B: VOICE-BASE XML FILES IN THE “EI SIRI”

i. General.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
<start>
    <block>Hello. What can I help you?
        <answer value="emergency" next="situation"/>
        <answer value="problem" next="situation"/>
        <answer value="accident" next="situation"/>
        <answer value="pain" next="situation"/>
        <answer value="sick" next="situation"/>
        <answer value="ill" next="situation"/>
        <answer value="trouble" next="situation"/>
        <answer value="pain" next="situation"/>
        <answer value="tightness" next="situation"/>
        <answer value="anxiety" next="situation"/>
        <answer value="lightheaded" next="situation"/>
        <answer value="weak" next="situation"/>
        <answer value="wheezing," next="situation"/>
        <answer value="vomite" next="situation"/>
        <answer value="vomiting" next="situation"/>
        <answer value="diarrhoea" next="situation"/>
        <answer value="cramps" next="situation"/>
        <answer value="difficulty" next="situation"/>
        <answer value="difficult" next="situation"/>
        <answer value="burned" next="situation_bs"/>
        <answer value="bernd" next="situation_bs"/>
        <answer value="burn" next="situation_bs"/>
        <answer value="burnt" next="situation_bs"/>
        <answer value="scalded" next="situation_bs"/>
        <answer value="sprains" next="situation_ss"/>
        <answer value="sprain" next="situation_ss"/>
        <answer value="sprained" next="situation_ss"/>
        <answer value="broken" next="situation_ss"/>
        <answer value="strains" next="situation_ss"/>
        <answer value="food poisoning" next="situation_fp"/>
        <answer value="asthma attack" next="situation_aa"/>
        <answer value="heart attack" next="ha#severe"/>
        <error statement="Please repeat it! If you know the situation, just say its name, otherwise, just say emergency."/>
    </block>
</start>
```
Okay! Goodbye! Thanks for using Emergency Intelligent Siri.

```
<end> Okay! Goodbye! Thanks for using Emergency Intelligent Siri.</end>
</root>

ii. situ_diag.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<situation statement="Okay, I need you to answer a few questions to determine the situation. Just tell me yes or no please.">
  <question id="1" statement="Is the patient having difficulty breathing?">
    <yes next="2"/>
    <no next="7"/>
  </question>
  <question id="2" statement="Is there chest pain and tightness?">
    <yes next="3"/>
    <no next="3"/>
  </question>
  <question id="3" statement="Does the patient have an overwhelming feeling of anxiety?">
    <yes next="4"/>
    <no next="5"/>
  </question>
  <question id="4" statement="Does the patient feel weak or lightheaded?">
    <yes next="ha#severe" statement="This patient may have had a heart attack."/>
    <no next="5"/>
  </question>
  <question id="5" statement="Does the patient have wheezing, which makes a whistling sound when they breathe?">
    <yes next="6"/>
    <no next="6"/>
  </question>
  <question id="6" statement="Has the patient had exposure to allergens and other triggers?">
    <yes next="situation_aa" statement="This patient may be having an asthma attack."/>
    <no next="situation_aa" statement="This patient may be having an asthma attack."/>
  </question>
  <question id="7" statement="Has the patient been vomiting?">
    <yes next="8"/>
    <no next="8"/>
  </question>
  <question id="8" statement="Does the patient have diarrhoea?">
    <yes next="9"/>
    <no next="9"/>
  </question>
  <question id="9" statement="Does the patient have stomach cramps?">
    <yes next="situation_fp" statement="This patient may have food poisoning."/>
    <no next="situation_fp" statement="This patient may have food poisoning."/>
```
A Mobile Speech Interface for Emergency Help: EI Siri

iii. level_diag.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<form id="level_diagnosis">
    <situation_fp statement = "Now, I need to check how severe the emergency is. Please only respond with yes or no."">
        <question id="1" statement = "Does the patient have changes in their mental state, such as confusion?">
            <yes next="fp#severe"/>
            <no next="2"/>
        </question>
        <question id="2" statement = "Does the patient have a high fever?">
            <yes next="fp#severe"/>
            <no next="3"/>
        </question>
        <question id="3" statement = "Is there blood in the patient's vomit or stools?">
            <yes next="fp#severe"/>
            <no next="4"/>
        </question>
        <question id="4" statement = "Does the patient have seizures or fits?">
            <yes next="fp#severe"/>
            <no next="5"/>
        </question>
        <question id="5" statement = "Does the patient have double vision or slurred speech?">
            <yes next="fp#severe"/>
            <no next="6"/>
        </question>
        <question id="6" statement = "Is the patient very dehydrated?">
            <yes next="fp#severe"/>
            <no next="fp#slight" statement="Don't worry. This does not seem to be a severe emergency!"/>
        </question>
    </situation_fp>
    <situation_ss statement = "Now, I need to check how severe the emergency is. Please only respond with yes or no."">
        <question id="1" statement = "Does the patient have bruising and swelling around a joint or muscle?">
            <yes next="ss#severe"/>
            <no next="2"/>
        </question>
    </situation_ss>
</form>
```
A Mobile Speech Interface for Emergency Help: EI Siri

Yanchao Yu (H00099161)

<situation_ss>

<question id="2" statement = "Can the patient move the injured joint?">
  <yes next="3"/>
  <no next="ss#severe"/>
</question>

<question id="3" statement = "Does the injured area look crooked or have lumps or bumps?">
  <yes next="ss#severe"/>
  <no next="4"/>
</question>

<question id="4" statement = "Can the patient put any weight on the injured joint or muscle?">
  <yes next="ss#slight" statement="Don't worry. This does not seem to be a severe emergency!"/>
  <no next="ss#severe"/>
</question>
</situation_ss>

<situation_aa statement = "Now, I need to check how severe the emergency is. Please only respond with yes or no.">

<question id="1" statement = "Are there severe and constant symptoms of wheezing, coughing, and a tight chest?">
  <yes next="aa#severe"/>
  <no next="2"/>
</question>

<question id="2" statement = "Is the patient too breathless to speak?">
  <yes next="aa#severe"/>
  <no next="3"/>
</question>

<question id="3" statement = "Is the patient's pulse racing?">
  <yes next="aa#severe"/>
  <no next="4"/>
</question>

<question id="4" statement = "Does the patient feel agitated or restless?">
  <yes next="aa#severe"/>
  <no next="5"/>
</question>

<question id="5" statement = "Are the patient's lips or fingernails blue?">
  <yes next="aa#severe"/>
  <no next="aa#slight" statement="Don't worry. This does not seem to be a severe emergency!"/>
</question>
</situation_aa>

<situation_bs statement = "Now, I need to check how severe the emergency is. Please only"
respond with yes or no."

<question id="1" statement = "Is the patient over 60 or under 5 years of age?"
  <yes next="bs#severe"/> 
  <no next="2"/></question>

<question id="2" statement = "Are there clothes stuck to the burn?"
  <yes next="bs#severe"/> 
  <no next="3"/></question>

<question id="3" statement = "Have the burns caused white or charred skin?"
  <yes next="bs#severe"/> 
  <no next="4"/></question>

<question id="4" statement = "Are these chemical or electrical burns?"
  <yes next="bs#severe"/> 
  <no next="5"/></question>

<question id="5" statement = "Is the burn bigger than the affected person's hand?"
  <yes next="bs#severe"/> 
  <no next="6"/></question>

<question id="6" statement = "Does the patient have burns on the face, hands, arms, feet, legs or genitals?"
  <yes next="bs#severe"/> 
  <no next="7"/></question>

<question id="7" statement = "Does the patient has heart, lung or liver disease, or diabetes?"
  <yes next="bs#severe"/> 
  <no next="bs#slight" statement="Don't worry. This does not seem to be a severe emergency!"/></question>
</situation_bs>
<error statement="Please just say yes or no!"/> 
</form>

iv. first_aid.xml

<?xml version="1.0" encoding="UTF-8"?>

<root>
  <severe statement = "That's really severe. I need to contact the Emergency Centre. Do you want to check your location?"
    <answer value="yes_se" para="location" next="block"/> 
    <answer value="no_se" next="block"/>
  </severe>
</root>
<block statement="Help is on the way, please wait patiently.">
</block>

<slight_ha statement = "I will provide some first aid information to help the patient:">
  <first_aid statement = "No.1 Make sure they are in a position that is comfortable for him or her. No.2 Give them constant reassurance while waiting for the ambulance.">
    <repeat statement = "Do you want me to repeat that?"
    <answer value="yes_sl" next="first_aid"/>
    <answer value="no_sl" next="end"/>
    </repeat>
  </first_aid>
</slight_ha>

<slight_ss statement = "I will provide some first aid information to help the patient:">
  <first_aid statement = "No.1 Apply an ice pack to the injury. No.2 Keep moving a sprained joint but immobilise a sprained muscle.">
    <repeat statement = "Do you want me to repeat that ?"
    <answer value="yes_sl" next="first_aid"/>
    <answer value="no_sl" next="end"/>
    </repeat>
  </first_aid>
</slight_ss>

<slight_bs statement = "I will provide some first aid information to help the patient:">
  <first_aid statement = "No.1 Stop the burning process as soon as possible. No.2 Remove any clothing or jewellery near the burnt area of skin. However, don’t try to remove anything that is stuck to the burnt skin because this could cause more damage. No.3 Cool the burn with cool or lukewarm water for 10-30 minutes. No.4 Keep the patient warm. No.5 Cover the burn with cling film. No.6 Treat the pain from a burn with paracetamol or ibuprofen."">
    <repeat statement = "Do you want to repeat it?"
    <answer value="yes_sl" next="first_aid"/>
    <answer value="no_sl" next="end"/>
    </repeat>
  </first_aid>
</slight_bs>

<slight_aa statement = "I will provide some first aid information to help the patient:">
  <first_aid statement = "No.1 Check whether the patient has any emergency medicine for an asthma attack. No.2 Help the person sit in a comfortable position and take their medication."">
    <repeat statement = "Do you want me to repeat that?"
    <answer value="yes_sl" next="first_aid"/>
    <answer value="no_sl" next="end"/>
    </repeat>
  </first_aid>
</slight_aa>