JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY (JKUAT)

DEPARTMENT OF COMPUTING

ICS 2406: COMPUTER SYSTEMS PROJECT

PROJECT SYSTEM ANALYSIS AND DESIGN REPORT

TITLE: AN IMPLEMENTATION OF SPEECH RECOGNITION IN AN EMERGENCY APPLICATION

AUTHOR:

NAME:	**NGIGI SHALON NJERI**	**REG. NO.:**	**CS281-1566/2014**
SUBMISSION DATE:	**_____**	**SIGNATURE:**	**_____**
COURSE:	**B. SC. COMPUTER SCIENCE**		
SUPERVISOR 1:			
NAME:	**PROF. WAWERU MWANGI**	**SIGNATURE:**	**_____**
DATE:	**_____**		
SUPERVISOR 2:	**SAMSON OCHINGO**		
NAME:	**_____**	**SIGNATURE:**	**_____**
DATE:	**_____**		

Table of Contents

1. INTRODUCTION

This report translates the requirements and processes into a technical design that will be used to develop the application. It covers system's methodology used, feasibility study, system requirements, and system analysis.

1.1 System Methodology

The project uses a mix of Extreme Programming (XP) Methodology and Rapid Application Development Methodology (RAD), where the main goal is to allow flexibility and quick adjustments to shifting requirements all the while lowering the cost of change in software requirements.

XP is a methodology that allows flexibility within the modelling process. The core principles of Extreme Programming revolve around Test Driven Development, Continuous Integration, Design Implementation, Small Releases, Simple Design, Code Standards and Conventions.

RAD is a process that proposes fast implementation of quality products, starting with development of preliminary data models and process models; which are then confirmed in the next stage using prototyping. Stages are repeated until a complete design is obtained.

The need for fast development and high quality results at a relatively low investment cost primarily make these methodologies a prudent choice [11].

1.2 Data Sources

PocketSphinx for android depends on a library called Sphinxbase that provides common functionality across all CMUSphinx projects which helps it support the following features through various APIs: [7]

- It uses abstract types which allow it to provide stability of source and binary compatibility as well
- It is fully re-entrant, so there is no problem having multiple decoders in the same process

- It has enabled a drastic reduction in code footprint and a modest but significant reduction in memory consumption

PocketSphinx consists of several components like knowledge base, decoder, language model and acoustic model. The language model, acoustic model and dictionary are EN-US files are downloaded from the CMUSphinx GitHub page [9].

PocketSphinx uses three models – an acoustic model (converts audio samples to phonemes), a language model (contains probabilities of sequences of words) and a phonetic dictionary (a mapping of words to phonemes).

2. FEASIBILITY STUDY

This is a study of how successfully the project will be completed subject to some factors affecting it. Factors within focus are technical, economic, operational, and schedule feasibilities.

2.1 Technical Feasibility

This looks into the availability of technical resources such as technical software development skills, software, and hardware to implement the project and use it successfully. The project requires knowledge in Java and Android Programming. It also requires an understanding of the speech recognition process.

Software requisites include an Android development environment (Android Studio), and a speech recognition engine (PocketSphinx). Hardware requirements include a computer, and a mobile phone with and android operating system for Testing. All of which are available, the project is therefore technically feasible.

2.2 Economic Feasibility

This weighs the cost of project development against its benefits, whether solving the problem is worthwhile. A speech driven application is important for situations when an individual is incapacitated and needs to call or ask for help. The application runs on Android Phones which are widely available. The cost of operating this application is almost negligible. The application will be running in the background waiting for a wakeup command. The development process however intensive since it requires learning while developing. The benefits of the project outweigh its costs.

2.3 Operational Feasibility

This feasibility looks into the whether the problem identified is relevant and useful to the users and or society. The application makes it possible to make to use a mobile phone hands free to make an emergency call or send an emergency text; for an incapacitated individual this app would be extremely helpful as it would also act as their medical IDs, containing information about their medical health such as allergies, current medication, and blood type. As such, it is operationally feasible.

2.4 Schedule Feasibility

This looks into the likelihood of the project being completed within reasonable and scheduled time. As per the set schedule the project is on track for timely completion. The analysis and design phase is within the schedule time. There are no foreseeable hitches that will delay the project.

3. SYSTEM REQUIREMENTS

3.1 Functional Requirements

This is a description of the system's function, services and operational constraints, what is defined should be implemented in the system. The application should allow for the following functional requirements [4]:

- The system shall be able to receive the sound information from microphone as data entry
- The voice recognition system shall be able to process the data entry from microphone
- The system shall be able to use the data entry from user to search for particular key words in the predefined database
- The system shall be able to save user profile and preset text message.
- The system shall be able to display the user profile, contacts list and settings
- The system shall be able to delete user profile
- The system shall be able to make a call to an operator.
- The system shall be able to send a text message to listed contacts.

3.2 Non-functional Requirements [12]

- **Reliability**

This refers to the ability of a component under stated conditions for a period of time. The application is expected to run in the background using minimal resources. The app should be activated or opened using a spoken keyword.

- **Accuracy**

The accuracy of recognition is important since the app is reliant on words and phrases to navigate through the application.

- **Availability**

Since the application is meant for use in emergency situations, it is expected to be readily available on the click of a button or a voice activation command.

- **Usability**

The User Interface of the application should be easy to use and navigate through.

- **Performance**

The app is expected to use minimal resources from the android OS and it shouldn't interfere or impede other apps within the phone.

3.3 User Requirements

These are requirements the user expects from software to be constructed in the project. The application should allow for the following functional requirements [6]:

- The user shall be able to activate the app using their voice
- The user shall be able to make a phone call to an operator using voice command "Call Operator"
- The user shall be able to send a preset text message to listed emergency contacts using voice command "Send SOS"
- The user shall be able to add, edit, and delete contacts in the contacts list.
- The user shall be able to edit their preset text message.
- The user shall be able to delete their profile
- The user shall be able to edit their profile.

4. SYSTEM DESIGN

4.1 Introduction

This section defines the architecture, components, modules, interfaces and data required for the system to satisfy specified requirements. The system design will be in terms of physical and logical design.

Physical design represents the physical entities of the software system, it provides information that is needed to build the system. It covers the following:

- Determining how a software system is packaged into its deployable units
- Determining which classes belong to which deployable units
- Managing the relationships between the deployable units

Logical issues include the determination of language constructs such as classes, operators, methods and packages. It involves arranging data into a series of logical relationships called entities and attributes, illustrated using Entity Relationship Diagrams (ERD).

4.2 Process Modelling

This is the process of representing how a system operates. Illustrating the activities being performed and how data moves among them [7].

4.2.1 Data Flows: Data Flow Diagram (DFD)

A Data Flow Diagram represents the abstract representation of the system by modelling processes, Entities, flow of data, relationships and storage of data. The diagrams below show different level DFDs.

DFD level 0

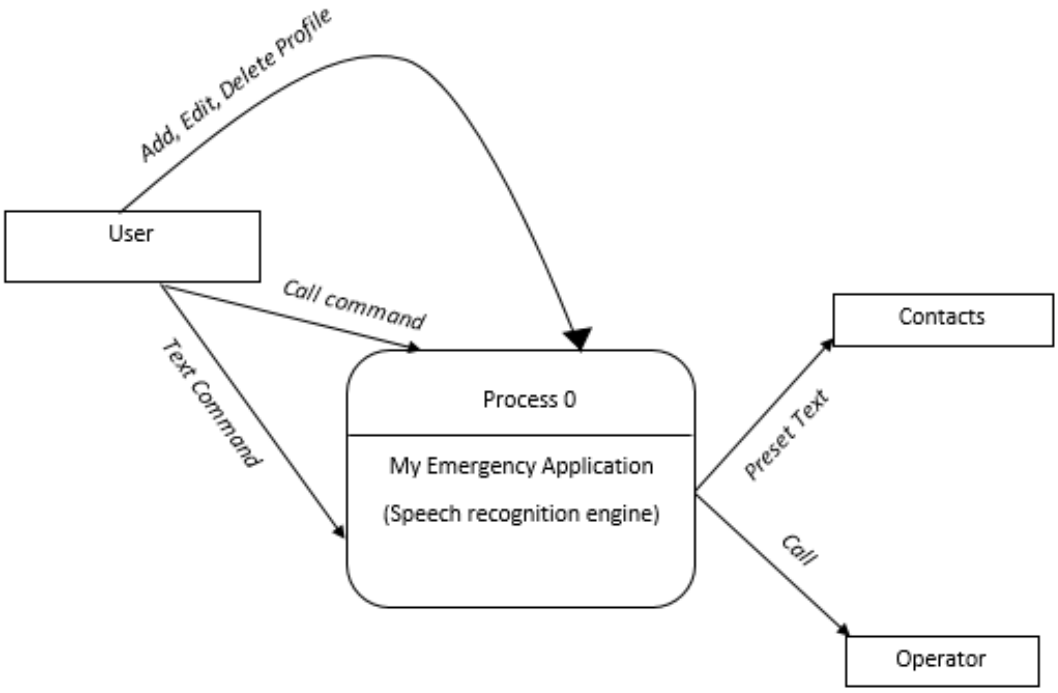


Figure 1.1: DFD level 0

DFD level 1

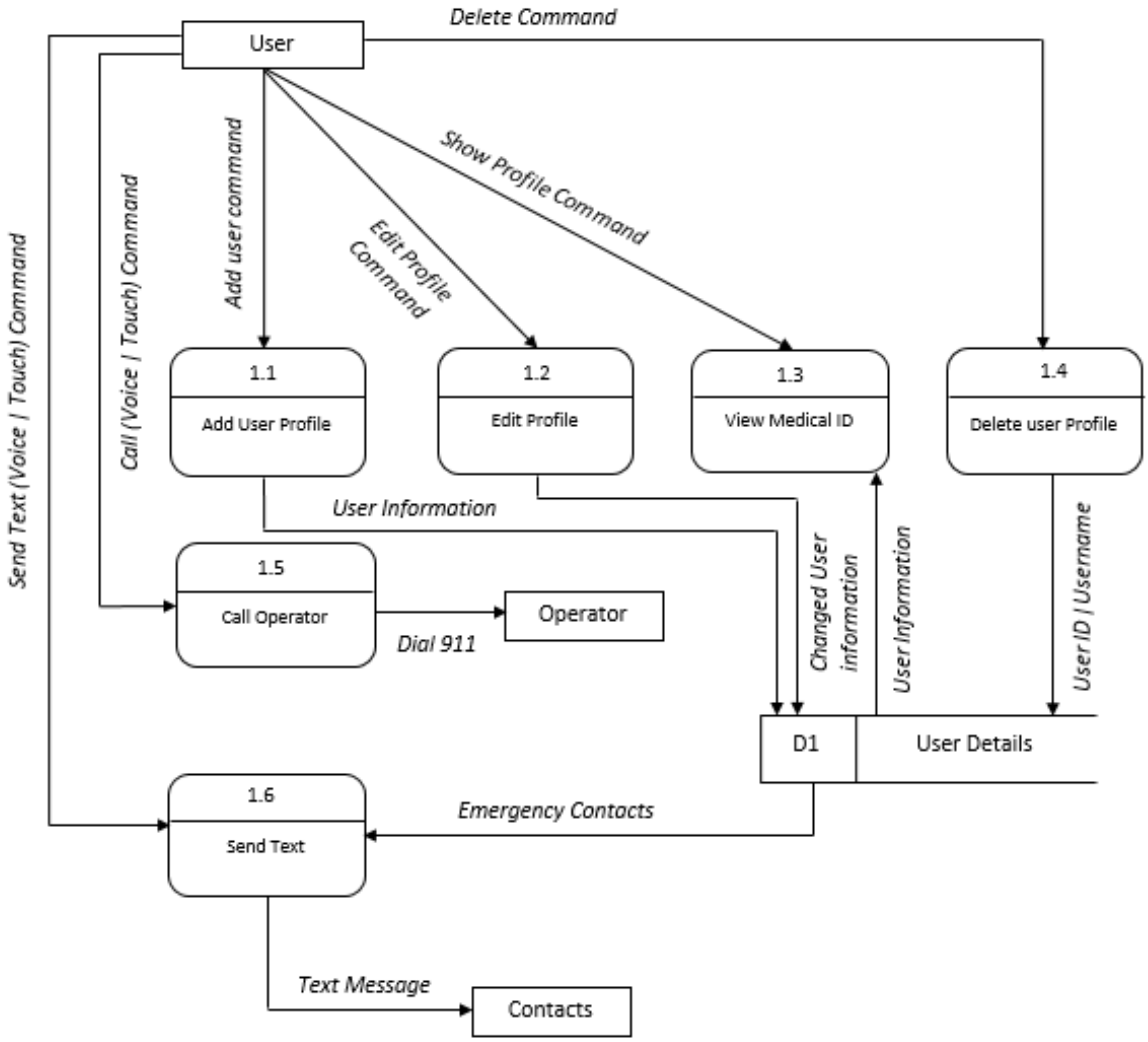
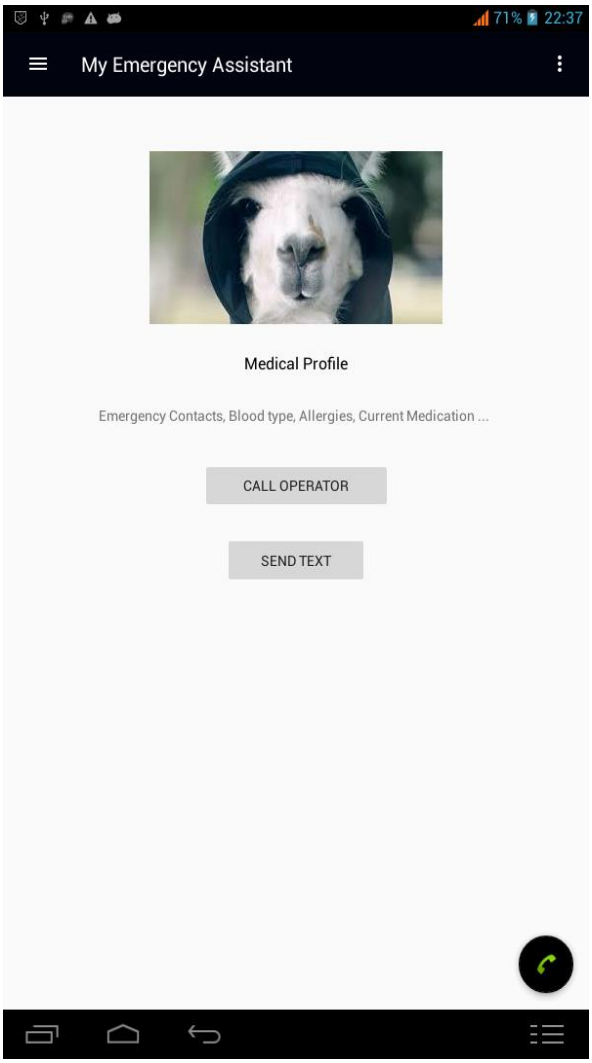


Figure 1.2: DFD Level 1

4.2.2 User Interface Design

Below are sample screenshots of the user interface [1].



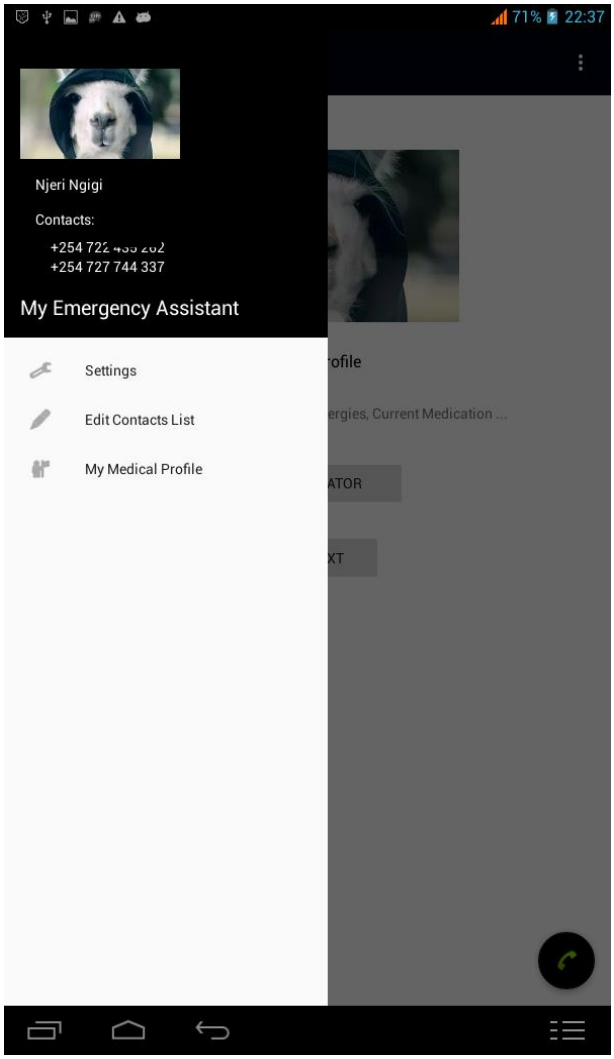
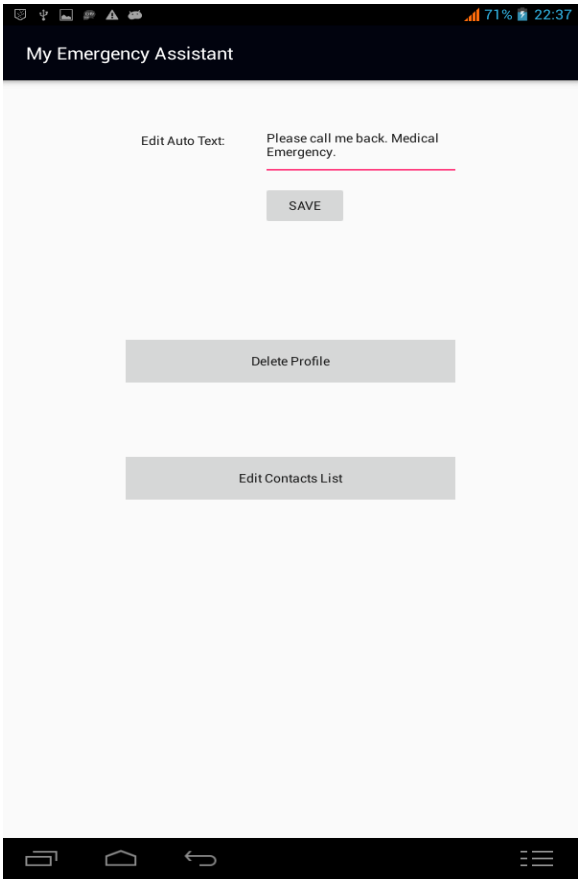


Figure 2.1 & 2.2:

The main activity which is the home with a drawable bar with links to the contacts list, settings and profile



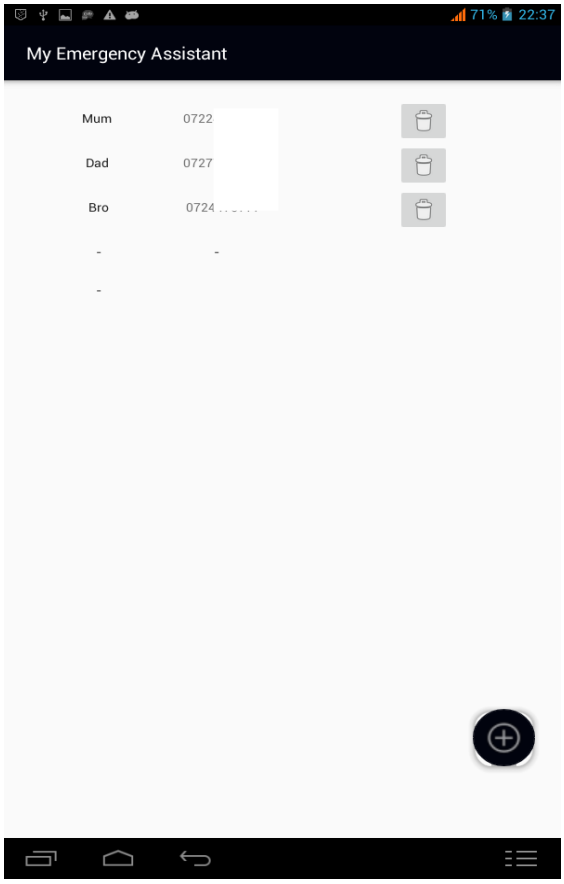


Figure 2.3 & 2.4:

The screenshots show the settings and contacts list respectively

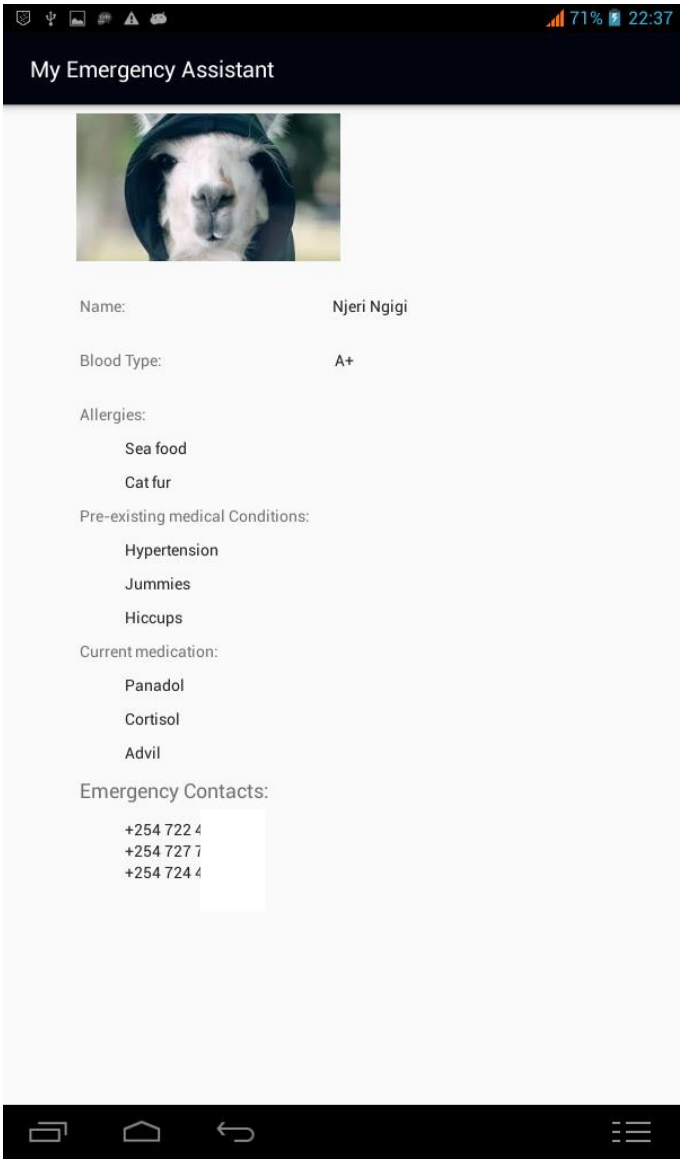


Figure 2.5:

User's Medical ID

4.2.3 Use Case Diagram

The use case diagram below shows the actors and the tasks they perform in relation to the system

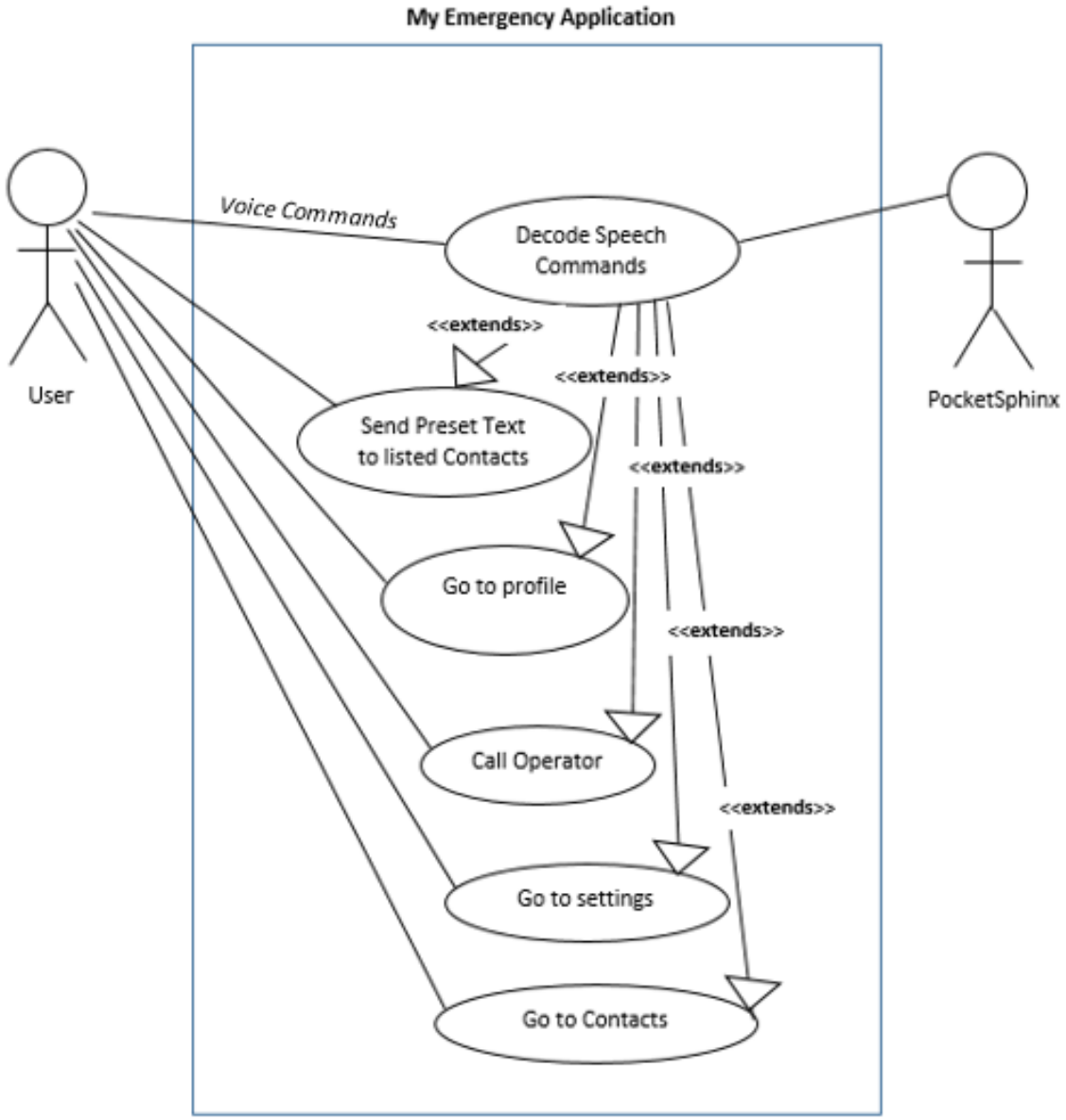


Figure 3: Use case Diagram

4.2.4 Class Diagrams

The diagram below shows the structure of classes used in the system and their relationships [3, 10].

The following classes were identified:

- **SpeechRecognizer** accesses the decoder functionality. This class is created with the help of SpeechRecognizerSetup builder, to configure decoder properties and parameters. SpeechRecognizer class has methods such as startListening () that starts the recognition process. The end of speech event can be obtained using onEndOfSpeech () callback of recognizer listener. The final result of recognition is produced in onResult () callback when recognizer.stop () is called.
- **edu.cmu.pocketsphinx.Assets#syncAssets** which synchronizes resources reading items from assets.lst file located on top of the *assets/.* It provides a method to automatically copy asset files to external storage of the target device.
- **DatabaseHelper** is a class that accesses and manages the SQLite storage. It contains methods such as onCreate (), onUpgrade (). It defines tables and columns, and provides methods for add, deleting and updating content in the database.
- **UserDetails class** defines a user object with fields: name, blood type, email, and phone numbers among others. It contains setter and getter methods for the fields.
- **Activity classes.** The project has at least 5 activity classes to setup and manage the User Interface. The classes have corresponding layout files to manage. Activity classes: StartActivity, MainActivity, ProfileActivity, SettingsActivity and ContactsActivity

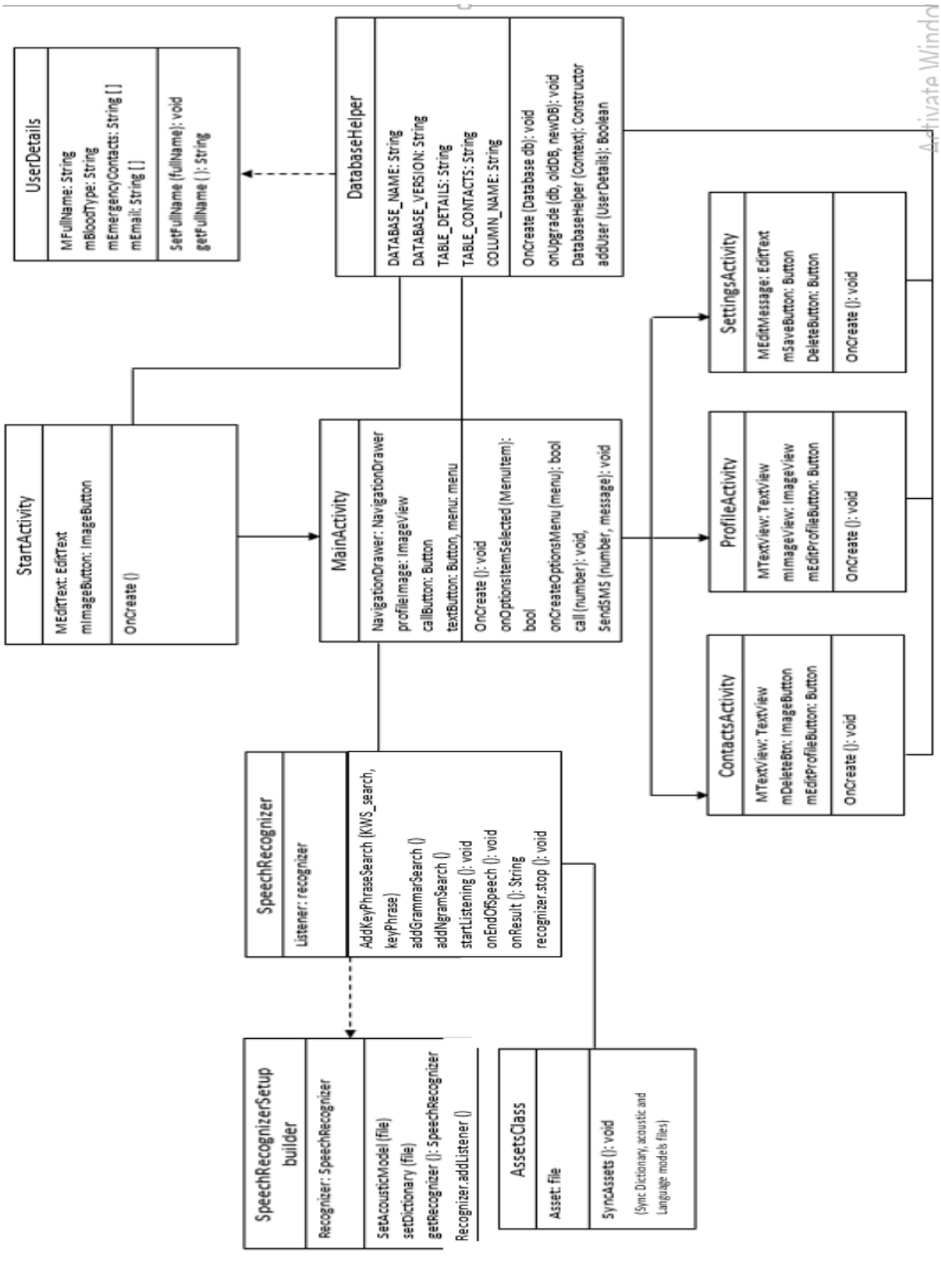


Figure 4: Class Diagram

4.2.5 Activity Diagrams

The activity diagram captures the dynamic behavior of the system. It shows the activity flow of the system, describes the sequence from one activity to another.

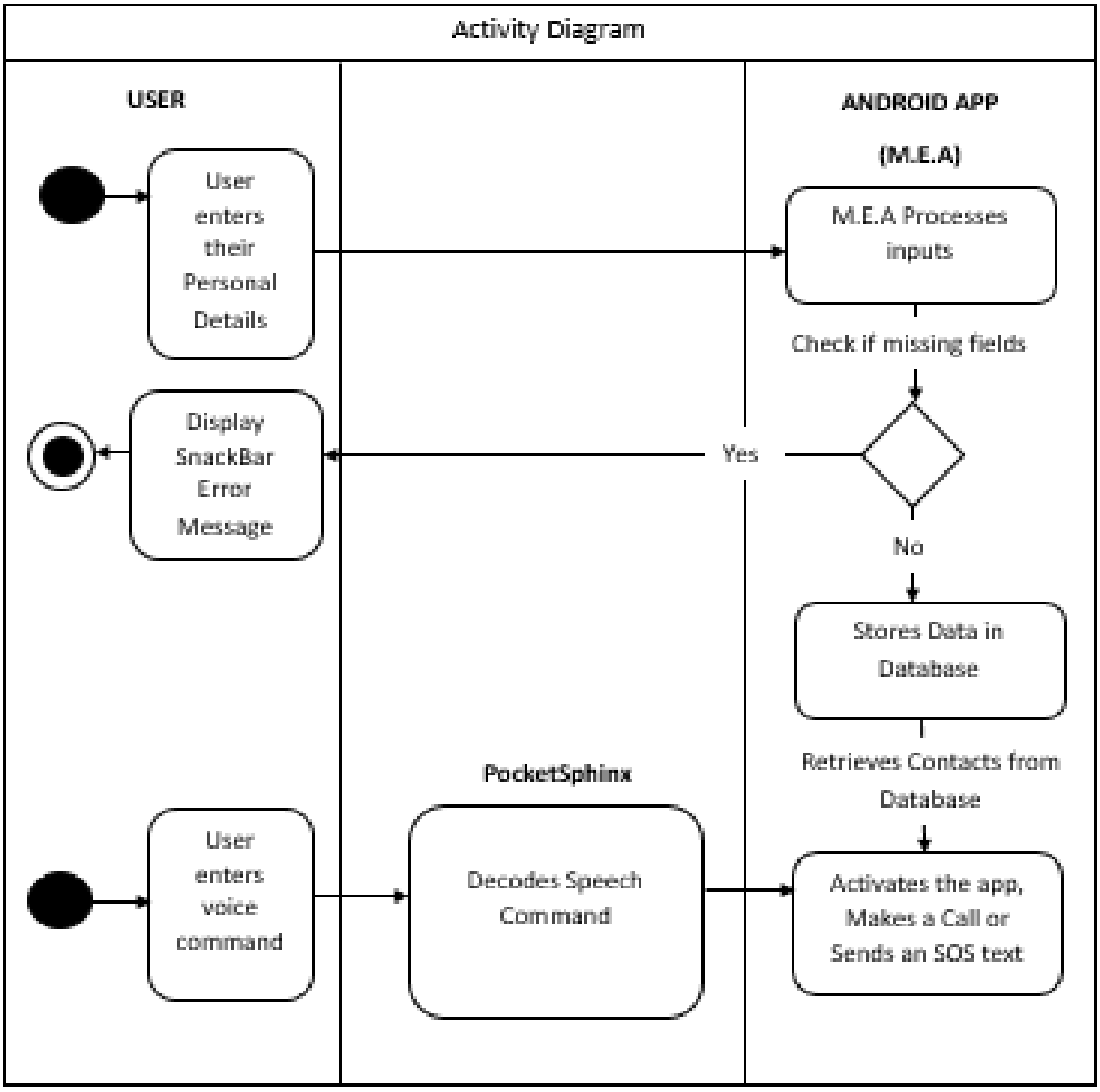


Figure 5: Activity Diagram

4.2.6 Database design and storage

The application uses SQLite database to store user details. It is not heavily dependent on storage since speech data is handled by the PocketSphinx Library.

The database has several tables:

User Details Table: This table contains user information with the fields – User's Full Name, User's Phone Number, email address, blood Type, allergies, medical conditions, current medication, and emergency contacts.

The primary key would be the name (FullName). However, Medical History, Allergies, Current Medication and Emergency Contacts would all be different tables that are referenced by User details table using foreign keys.

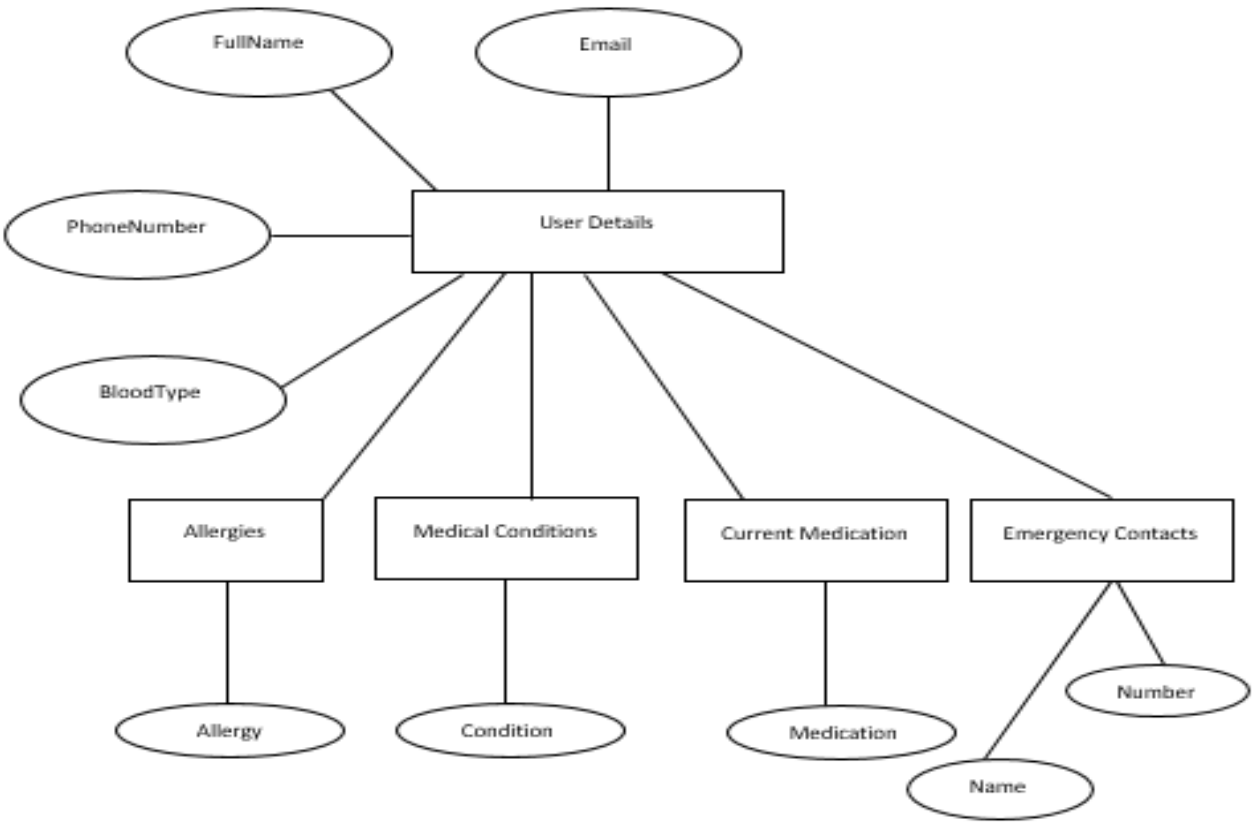


Figure 6: Entity Relationship Diagram for user details stored

REFERENCES

1. Bill Philips, Chris Stewart, Kristin Marsicano*, Android Programming: The Big Ranch Guide*, 3rd Edition, 2017, Pearson Technology Group.
2. Ben James Kraal. Considering Design for Automatic Speech Recognition in Use. (April 2006). University of Canberra ACT 2601, School of Information Sciences and Engineering, B. SC. Information Technology (Hons).
3. Pooja V. Janse, Ratnadeep R. Deshmukh. (Sept. – Oct. 2014). Design and Development of Database and Automatic Speech Recognition System for Travel Purpose in Marathi. IOSR Journal of Computer Engineering (IOSR_JCE). *e-ISSN: 2278-0661,p-ISSN: 2278-8727*
4. Iv´an Francisco, Castro Cer´on, Andrea Graciela, Garcia Badillo. (June, 2011). A Keyword Based Interactive Speech Recognition System for Embedded Applications. School of Innovation, Design and Engineering M¨alardalen University V¨aster˚as, Sweden.
5. Mark Knight. (2004/ 2005). Analysis, Design and Implementation of a Helpdesk Management System Information Systems (Industry).
6. Paul Lamere, Philip Kwok, William Walker. (2003).Design of the CMU sphinx-4 decoder. Sun Microsystems Laboratories, Carnegie Mellon University, Mitsubishi Electric Research Laboratories, USA.
7. PocketSphinx Android demo; using pocketsphinx-android, June 2018. [https://cmusphinx.github.io/wiki/tutorialandroid/]
8. My Droid Experience, February 2011. Offline Speech Recognition with PocketSphinx. [http://swathiep.blogspot.com/2011/02/offline-speech-recognition-with.html]
9. SpeechRecognition 3.2.0, February 2016. [https://pypi.org/project/SpeechRecognition/3.2.0/]
10. System Methodology: RAD & Extreme Programing, June 2018. [http://www.itinfo.am/software-development-methodologies/]
11. Non-functional Requirements, March 2018. [https://en.wikipedia.org/wiki/Non-functional_requirement]