METU-Computer Engineering

TETRIS TEAM

SMART DRIVER ASSISTANT

SOFTWARE DESIGN DESCRIPTIONS

Team Members:

Seymur Mammadli
Shkelim Memmola
Nail Ibrahimli
Mehmet Kurhan

PREFACE

This Document contains the system design information about SMART DRIVER ASSISTANT. This document is prepared according to the "IEEE Standard for Information Technology – Systems Design – Software Design Descriptions – IEEE Std 1016 – 2009". This Software Design Documentation provides a complete description of all the system design and views of the project. The first section of this document includes Project Identification, Stakeholders Identification and requirements, Composition of the developers' team. The following sections include document purpose and design viewpoints of the system.

Table of Contents

Tables

Figures

1. Overview

This document contains the software design descriptions for SMART DRIVER ASSISTANT project. It is prepared according to the "IEEE Standard for Information Technology – Systems Design – Software Design Descriptions – IEEE 1016 – 2009". The Software Design Descriptions Document is a document where the information about how the system should be built is provided. The details are represented by using graphical notations such as use case models, sequence diagrams, class diagrams, object behavior models, and other supporting requirement information.

1.1 Scope

This document contains a structural overview of all modules, interfaces and data. It also covers a detailed design of each module by giving information about the overall software architecture and the design methods for each module of the software project. A set of design views will be presented in order to support the design and development process. This document will serve as a guideline through the implementation phase.

1.2 Purpose

The purpose of this document is to describe and visualize the design and architecture of SMART DRIVER ASSISTANT by using different viewpoints. Its aim is to describe the software system which is structured to meet the needs specified in Software Requirements Specification document. This document will be the primary reference for the implementation phase.

1.3 Intended audience

The expected audience for this document is the design and development team of the SMART DRIVER ASSISTANT project for implementing purposes.

2. Definitions

IEEE: Institute of Electrical and Electronics Engineers

GUI: Graphical User Interface.

User: A class for people after logging in.

Message: Objects we used for communications between server and client

SRS: System Requirements Specification

EDS: Eye Detection System

UML: Unified Modeling Language

TTS: Text To Speech

STT: Speech To Text

CPU: Central Processing Unit

SDK: Software Development Kit

SDD: Software Design Description

SRS: Software Requirements Specification

MVC: Model View Controller

3. Conceptual model for software design descriptions

Basic terms, concepts and context of SDD will be given in this section.

3.1 Software design in context

This project will be designed as object – oriented in a modular design fashion. This will enable code extensibility and reusability.

Android native SDK will be used. The project will be implemented with Java language using Android Studio framework. Additionally any operating system can be used. Google's libraries will be extensively used.

3.2 Software design descriptions within the life cycle

3.2.1 Influences on SDD preparation

The key software life cycle product that drives a software design is typically the software requirements specification. The requirements in the SRS (product perspective, functional and non-functional requirements and interface requirements) and the opinions gathered from individual researches, project advisor and team members, influenced in the preparation of this document.

3.2.2 Influences on software life cycle products

This project consists of three big parts – voice commanding, crash detection and drowsy driver detection. The agile development will be used throughout all the project, and in the first cycle, the consistency and the correctness of each part of the project is expected.

The feedbacks from stakeholders will be taken into the consideration and that will lead in planning new iterations and modifying existing requirements into new ones.

During the second cycle necessary module changes will be performed and adapted. After Demo Day, all the feedbacks that will be taken from different sources will be further analyzed for further product cycles. Then project will be ready for the market.

3.2.3 Design verification and design role in validation

Verification and validation will be tested after preparation of the test cases. All system parts will be tested against these cases. It will be checked for whether the requirements specifications are fulfilled or not.

4. Design description information content

4.1 Introduction

The aim of SDD in this section is to provide information about how the design description will be explained in the later sections of this document.

Throughout the document identification, diagrams, user views and user viewpoints are provided.

4.2 SDD identification

This is the initial SDD for the project Smart Driver Assistant of the development team. The date of issue for this document is: 03.06.2016.

The issuing organization for this document is team “TETRIS”

The supervisors of this project are Dr, Atilla Ozgit, Assoc. Prof. Emre Akbas, Assoc. Prof. Ahmet Oguz Akyuz, and Assistant Huseyn Aydin.

The scope of this document is the software design information about the project Smart Driver Assistant. In this design report, UML will be mainly used UML for explaining the design viewpoints.

4.3 Design stakeholders and their concerns

The design stakeholders of this project are TETRIS team. Other stakeholders are instructors of Computer Engineering Design course Prof. Dr. Atilla Ozgit and Assoc. Prof. Emre Akbas, project supervisor Assoc. Prof. Ahmet Oguz Akyuz, and course assistant Huseyn Aydin.

Smart Driver Assistant targets all car drivers. As for the voice commanding part, due to the background noise, a robust speech evaluation is expected. As for crash detection part, due to road conditions, a robust sensor detection measure and handler is required. And as for the eye detection part, due to low light conditions, a robust closed eye detector is expected. And this is what TETRIS team is mostly concerned.

For the stakeholders, team meetings and team progress are their main priority concerns.

4.4 Design views

This project will be implemented as modular structure. MVC architecture will be used as architecture pattern. Object-oriented design is chosen for this project. This give the advantage of integrating new features to Smart Driver Assistant project easily and modifying the existing components.

Product context is specified and all restricted limitations are given in the SRS document before. Composition and team organization is equally made. Design views shows estimated cost, staffing, documenting and scheduling. A logical view of the product is explained and also it is supported by diagrams. Relationships of the classes are easily perceived. Each used view will be presented with its viewpoint and with their description as well.

4.5 Design viewpoints

Each viewpoint used are described as below:

Context viewpoint: shows what is expected from the user actor in the system. The roles of the user and stakeholders are clearly specified.

Logical viewpoint: shows the logical class structure of MVC pattern.

Composition viewpoint: is used for giving a higher level view to the system.

Information viewpoint: is the viewpoint which explains the structure of data about the system.

Pattern use viewpoint: is used for showing the architectural design of the system

Interaction viewpoint: is explaining the interactions between modules in the system like interactions between modules.

Dynamic viewpoint: is used for explaining the system in a state described way.

4.6 Design elements

Design elements will be explained in details in section 5.

4.7 Design rationale

Design choices are made according to some significant features like sustainability, integrating another project. It can be update-able according to stakeholders and users requirements. Each function in the software will be commented so that it can be understandable for the other developers and also they can change the code by help of these comments MVC is chosen as architecture design pattern because MVC architectural pattern separates an application into three main components: the model, the view, and the controller. The data is controlled in model, and views via controller. That makes easy to change information in tables.

4.8 Design languages

UML is selected as a part of design viewpoint specification.

5 Design viewpoints

5.1 Introduction

In this part, seven main design viewpoints will be explained.

- Context viewpoint
- Logical viewpoint
- Composition viewpoint
- Information viewpoint
- Pattern use viewpoint
- Interaction viewpoint
- State dynamics viewpoint

During the explanation of these viewpoints, UML diagrams will be used to increase understandability

5.2 Context viewpoint

Context viewpoint contains the relationships, interactions and dependencies between the system and the environment. Context diagram and use case diagram are presented to make a picture of context viewpoints by showing the relationships and interactions between the system and user of system.

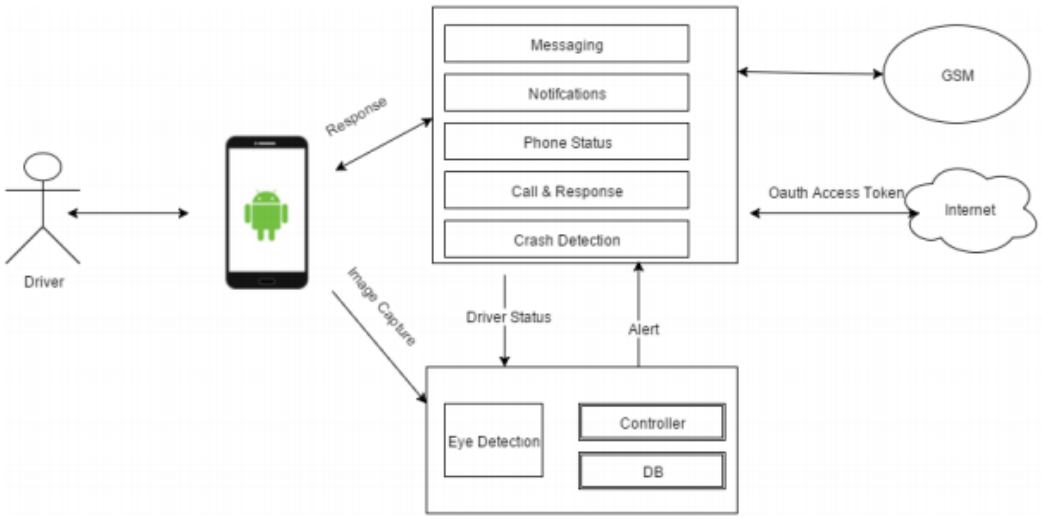

Figure 1: Context Diagram of Smart Driver Assistant

From the figure it is obvious that Smart Driver Assistant execute the commands which mostly requested by user's voice. The following commands will be implemented:

- Call
- Send a message
- Read out a message
- Read out incoming notifications
- Recording video

- Activate safeguard mode

Our products also detects Car crashed and asks for feedback from the driver. In the case of negative feedback or no feedback emergency message with the user's location information is sent to predefined contact number.

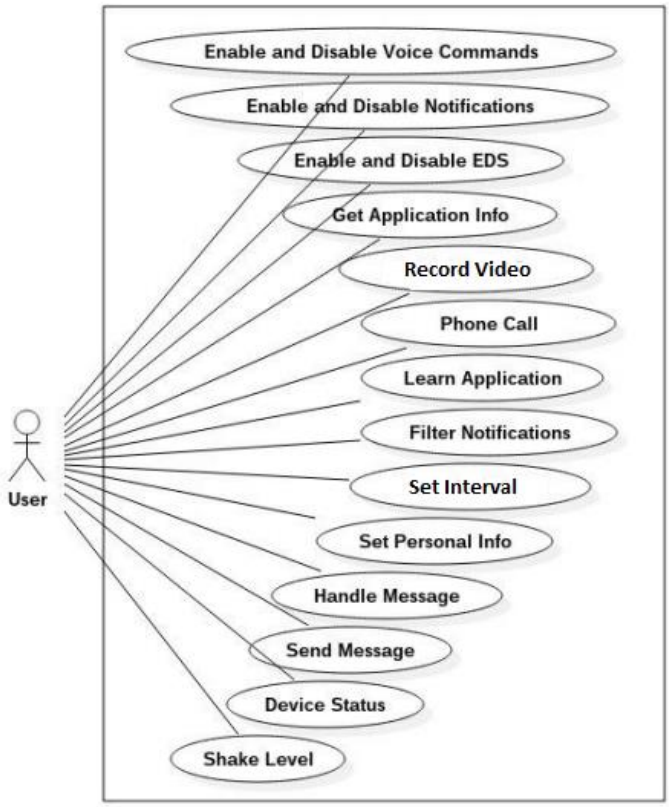

Figure 2: Use case diagram of Smart Driver Assistant

The system will perform the following use cases and relations:

Enable and Disable Voice Commands: User can enable and disable voice commands

Enable and Disable Notifications: User can enable and disable notifications

Enable and Disable EDS: User can enable and disable Eye Detection System

Get Application Info: User can get general information about the application

Record Video: User can record the video

Phone Call: User can perform phone call

Learn Application: User can learn how the application works

Filter Notifications: User can filter notifications

Set Interval: User can adjust time interval for video record

Set Personal Info: User can set his/her name and emergency phone number

Handle Message: User can handle an incoming message

Send Message: User can send a message

Device Status: User can acquire device status

Shake Level: User can set the shake level

5.3 Composition viewpoint

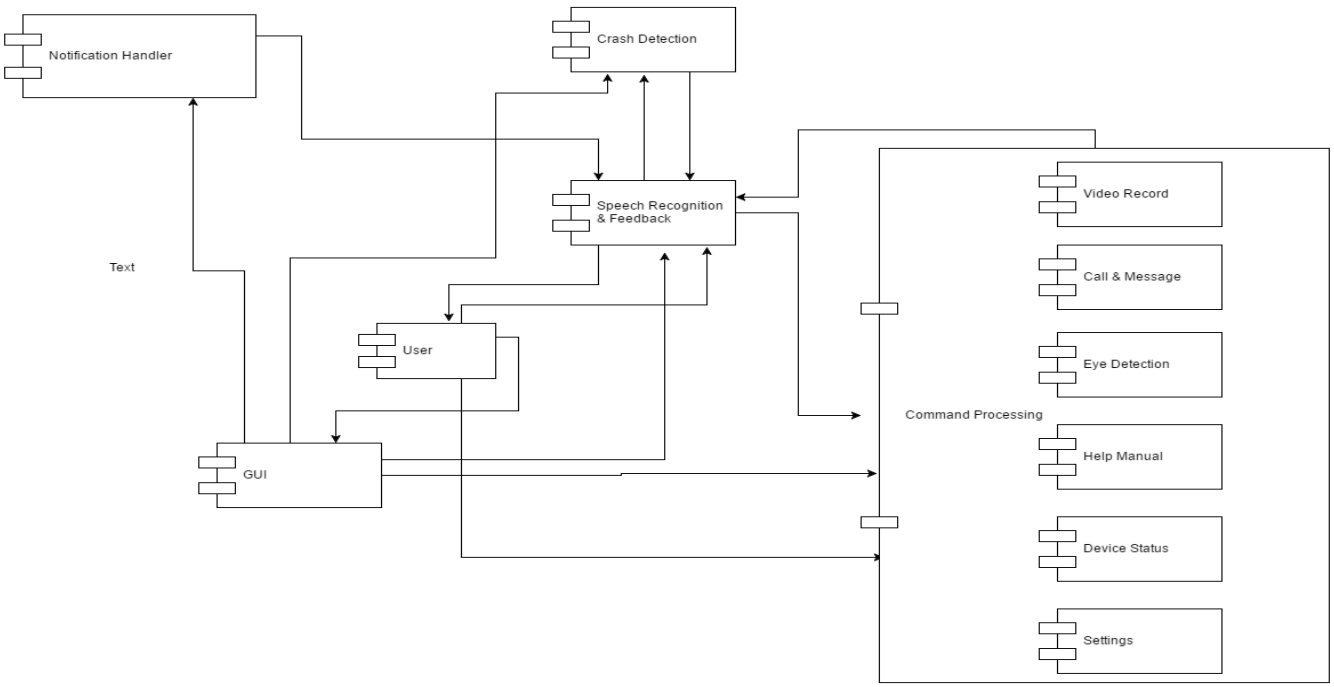

Figure 3: Composition diagram

This viewpoint focuses on the structure of the system and provides a top level view of the entire system from the perspective of each component. For this aim, for the logical representation UML Component Diagram is provided as it can be seen from the figure

Speech to text and text to speech used mostly for interaction between user and application.

GUI can also be used for enabling and disabling features and for doing some other operations mostly about the adjustment in setting the time interval of video record.

5.4 Logical viewpoint

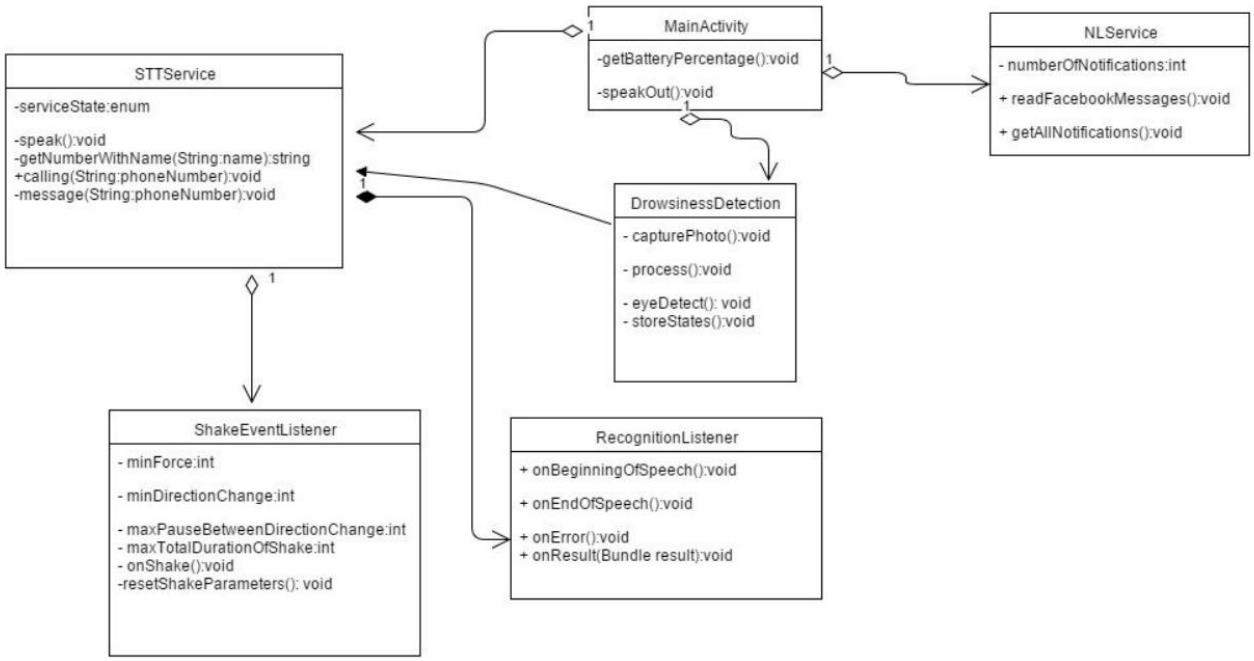

Figure 4: Class Diagram

This viewpoint contains the static structure of system with entities, interaction among entities and software patterns that applied to the static structure of the system. Figure shows associated class diagram which contains information about logical viewpoint.

More information about classes and their fields and methods can be found in the given table.

Main Activity	This class is the starting point of our project. It starts all the services. **Methods** getBatteryPercentage(): Gets the percentage of battery level. SpeakOut(): Triggers Text-To-Speech
STTService	This class is responsible for Speech-To-Text. It is supposed to work at background all the time and receive voice commands. **Attributes** serviceState: Stores the current state of service`: WAITING, LISTENING, PROCESSING **Methods** getNumberWithName(String name): Gets the phone number of according to the name from contacts. calling(String phoneNumber):Calls the number. speak(): Triggers Text-To-Speech message(String phoneNumber): Sends message.
ShakeEventListener	This class implemented for detecting car crashes. **Attributes** minForce : Minimum force amount for detecting crash. minDirectionChange: Minimum direction change…

	maxPauseBetweenDirectionChange: Maximum pause maxTotalDurationOfShake: Duration of shake **Methods** onShake(): Callback function for shake event. resetShakeParameters():Resetting the parameters for shake.
RecognitionListener	This class implemented for speech recognition as background. **Methods** onBeginningOfSpeech():Callback function for beginning of speech. onEndOfSpeech():Callback function for end of speech. onError():Callback function for errors. onResult(Bundle result): Callback function for result
NLService	This class implemented for notification handling. **Attributes** numberOfNotifications: Number of notifications that waits. **Methods** readFacebookMessages(): Reads incoming facebook messages. getAllNotifications(): Getting all notifications.

DrowsinessDetection	This class implements driver drowsiness detection **Methods** capturePhoto():Taking photos of driver. process(): Process states and make a decision. eyeDetect():Detecting the state of eyes. storeStates():Stores the states

Table 1: Class fields and methods

5.5 Information viewpoint

Information viewpoint contains the system's storage, management and manipulation of data and the persistent data information.

Application does not need large amount of memory for storing the data. Since it is a mobile application, we were obligated to process the data just using small amount of memory and waste small percentage of CPU usage.

While implementing Driver Drowsiness Detection all frames were not stored. Each frame was just captured processed and released. States were stored as flags. TinyDB were used for keeping small amount of data like user preferences.

5.6 Pattern use viewpoint

Pattern use viewpoint addresses design ideas as collaboration patterns involving abstracted roles and connectors. The pattern which is used in our project is Model View Controller (MVC) pattern. In this pattern, user will interact with the system by a user interface and by means of

voice commands. When user makes an action like commanding or changing the settings controller start handling the command. Speech to Text library was integrated to get the voice command. The controller notifies the model of the user interaction. Then a view manages the display of information and interacts with user .Text to Speech library was integrated to give the feedback to user about the result of interaction.

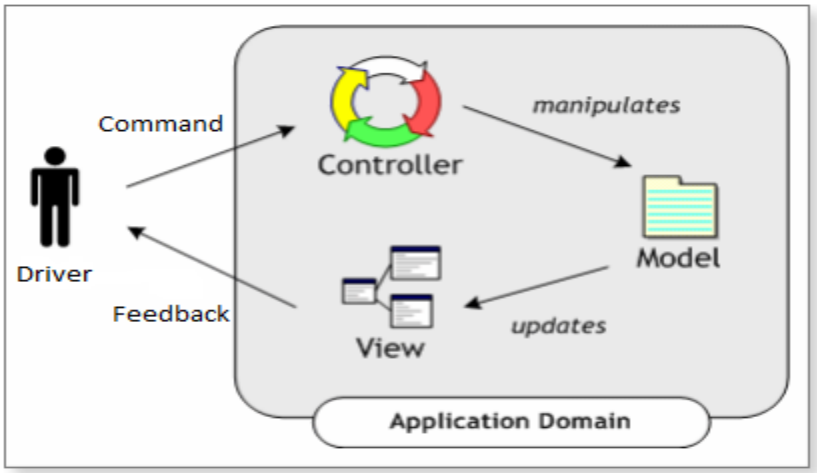

Figure 5: Model View Controller pattern

5.7 Interaction viewpoint

In this section the interaction among entities of the system will be visualized. As a design tool UML Sequence Diagrams are used in order to provide representation of the interaction.

5.7.1 Design concerns

Smart driver assistant can work with or without internet. The application also uses the GSM module of the phone. Voice commands can only work with the internet. The interaction between the user and the voice command system is shown below with the help of sequence diagrams.

5.7.2 Voice commanding system

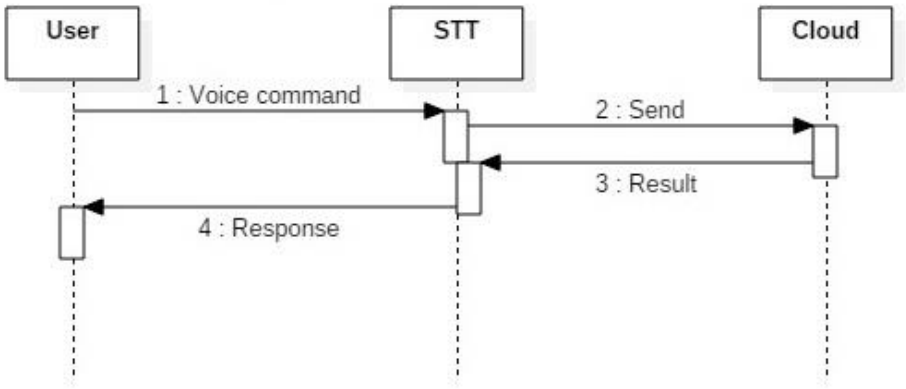

Figure 6: General overview of voice commands system

The above figure shows the interaction between user and voice commanding system. The STT service gets the voice commands from the user and send them to the cloud service for analyzing. After the STT fetches the results from the cloud, it gives response to the user accordingly.

5.7.3 Call Command Interaction

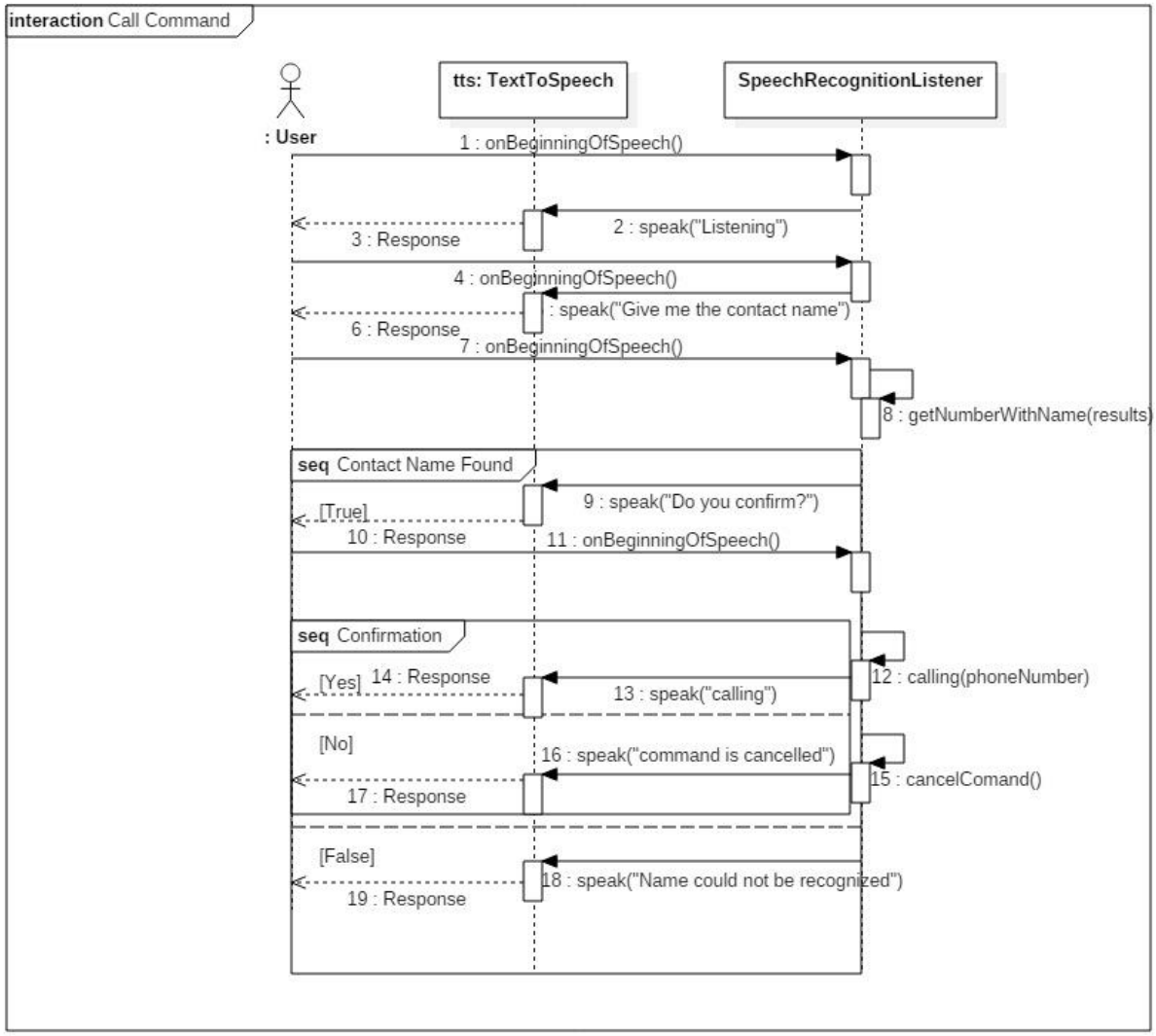

Figure 7: Call command sequence diagram

The diagram above shows the call interaction between the user and the application. First, user says the starter command and SpeechRecognition service enters the listening state. Then, the user says the call command. After that, service starts to wait for the contact name. If the contact name said by the user is found, the application asks for confirmation, else the application asks for the contact name again. In case of the user confirmation the message, the application calls the contact's number. If the user does not confirm the message, the command is cancelled.

5.7.4 Message Command Interaction

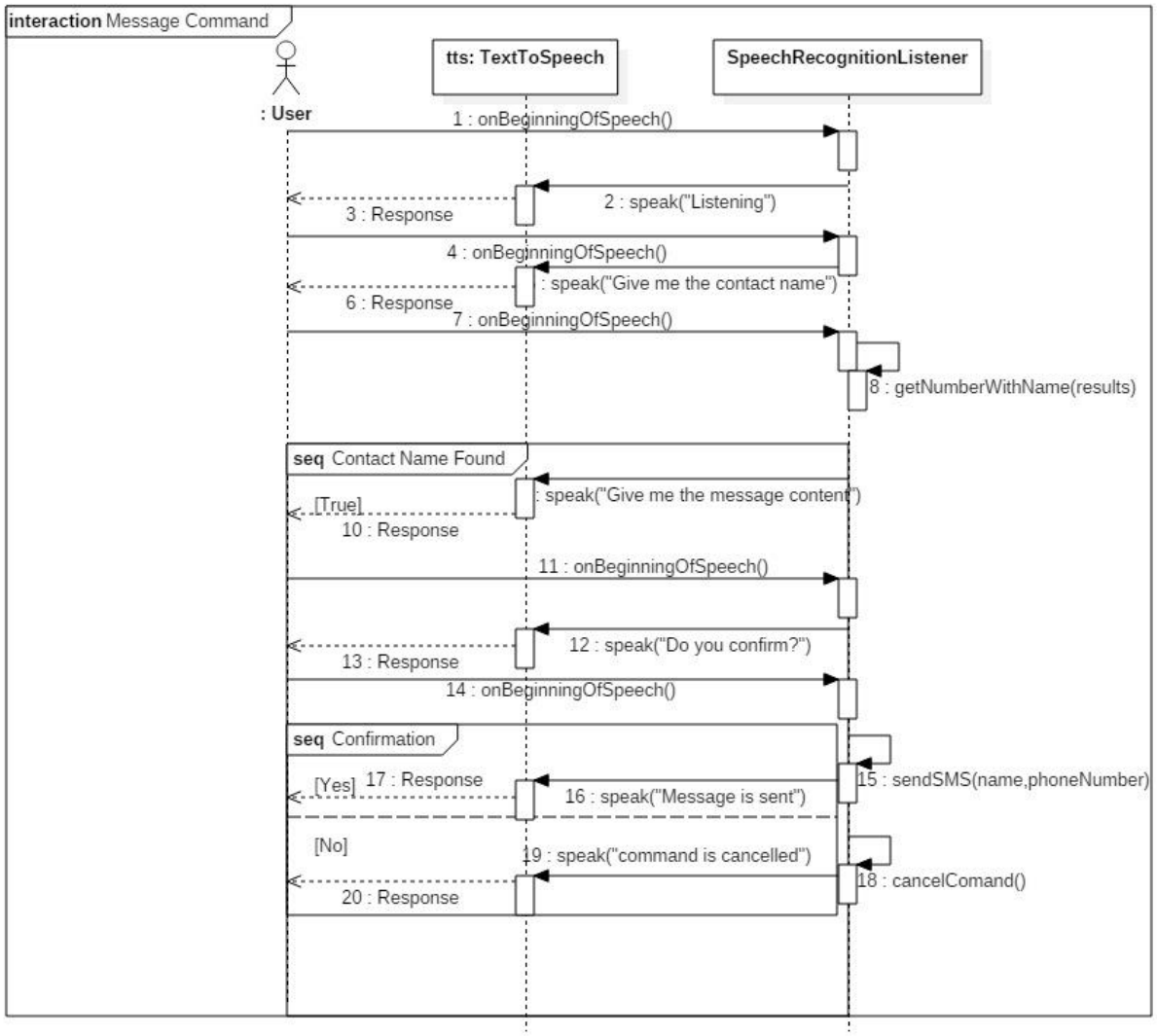

Figure 8: Message command sequence diagram

The diagram above shows the message interaction between the user and the application. First, user says the starter command and SpeechRecognition service enters the listening state. Then, the user says the message command. After that, service starts to wait for the contact name. If the contact name said by the user is found, the application requests the message content, else the application asks for the contact name again. After the user says the message content, the application waits for confirmation. If the user confirms the message, message is sent to the contact. If the user does not confirm, the command is cancelled.

5.7.5 Date Command Interaction

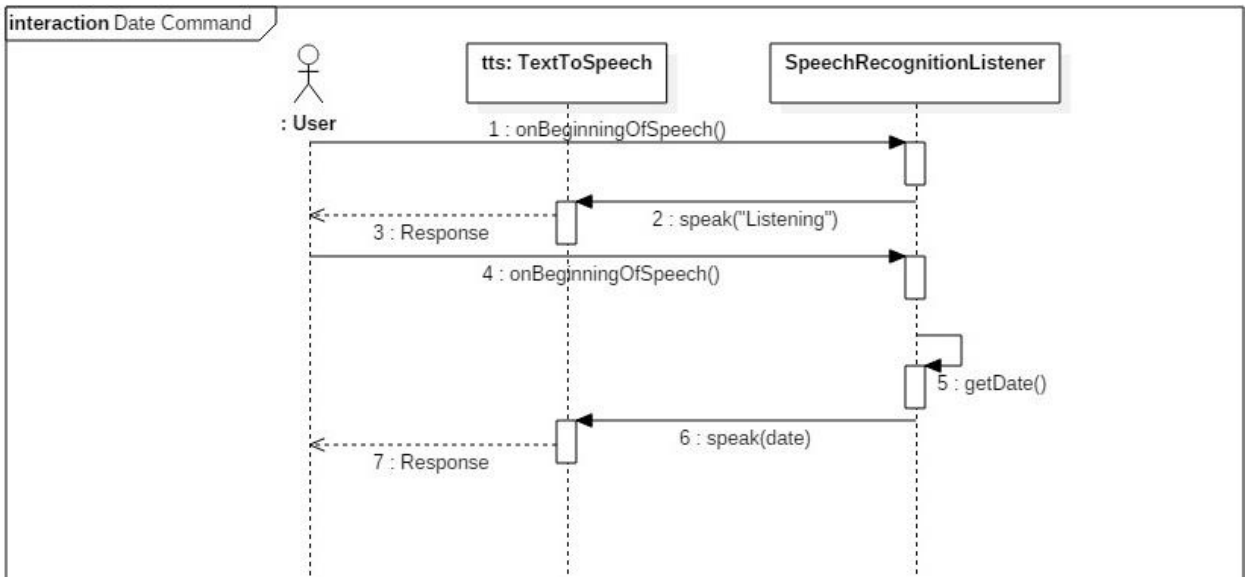

Figure 9: Date command sequence diagram

The figure above shows the date command interaction between the user and the application. Initially, user says the starter command and the service enters the listening state. Then user says the date command. The application gets the current date and responds to the user using TTS engine.

5.7.6 Time Command Interaction

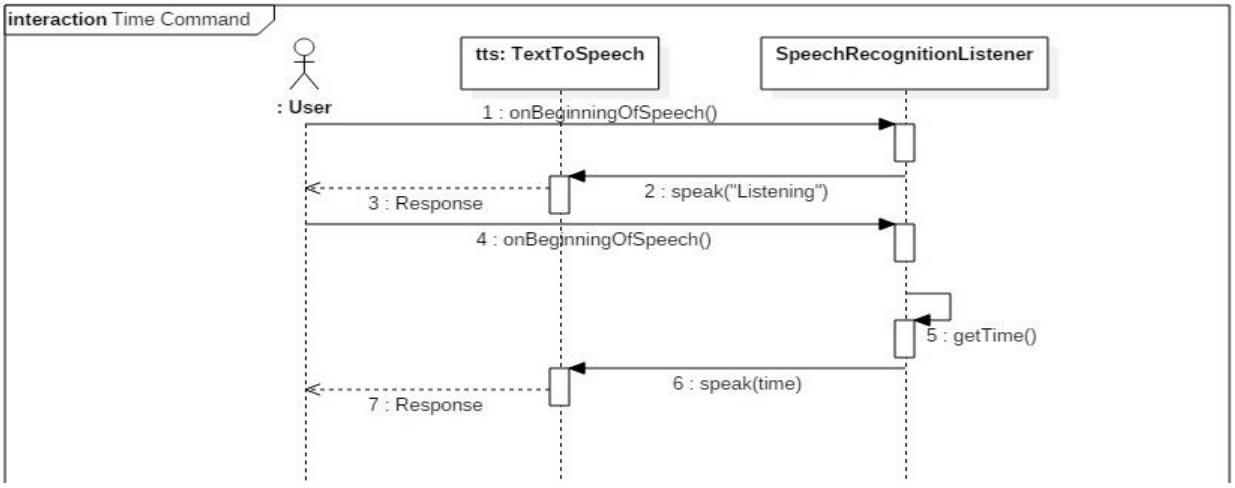

Figure 10: Time command sequence diagram

The figure above shows the time command interaction between the user and the application.

Initially, user says the starter command and the service enters the listening state. Then user says the time command. The application gets the current time and responds to the user using TTS engine.

5.7.7 Battery Command Interaction

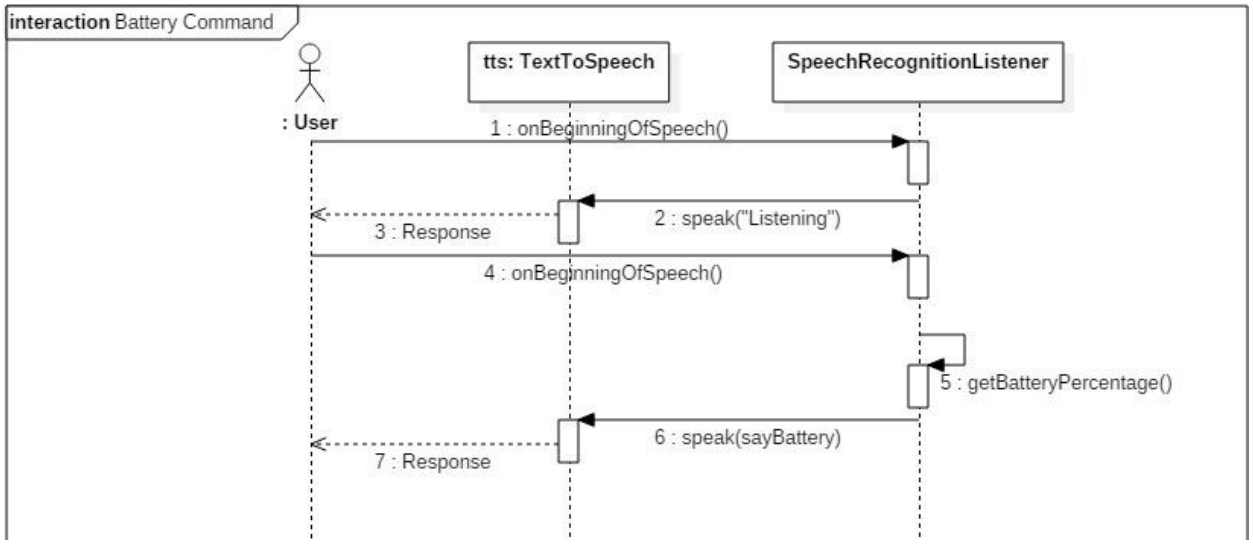

Figure 11: Battery command sequence diagram

The figure above shows the battery command interaction between the user and the application. Initially, user says the starter command and the service enters the listening state. Then user says the battery command. The application gets the current battery level and responds to the user using TTS engine.

5.7.8 Detect Command Interaction

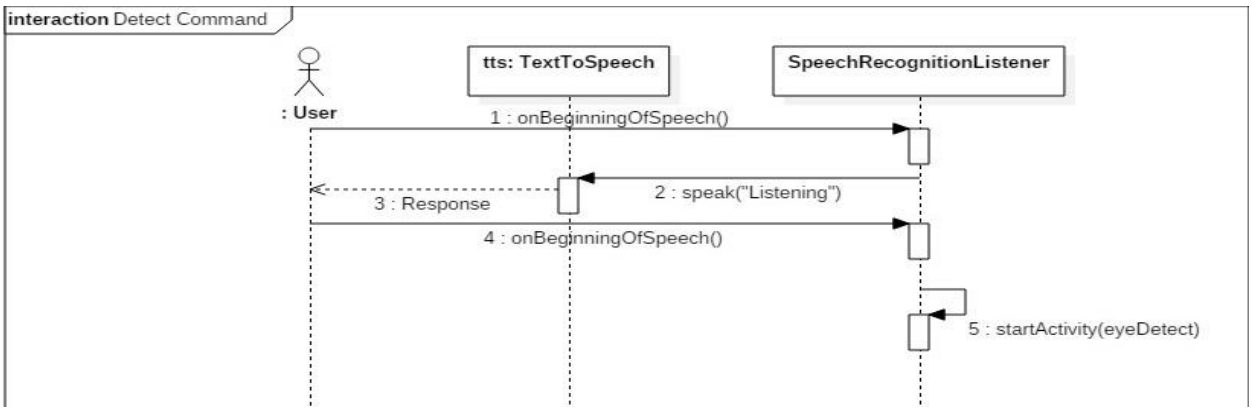

Figure 12: Detect command sequence diagram

The figure above shows the battery command interaction between the user and the application. Initially, user says the starter command and the service enters the listening state. Then user says the detect command. After that, the application starts the eyeDetect activity.

5.7.9 Record Command Interaction

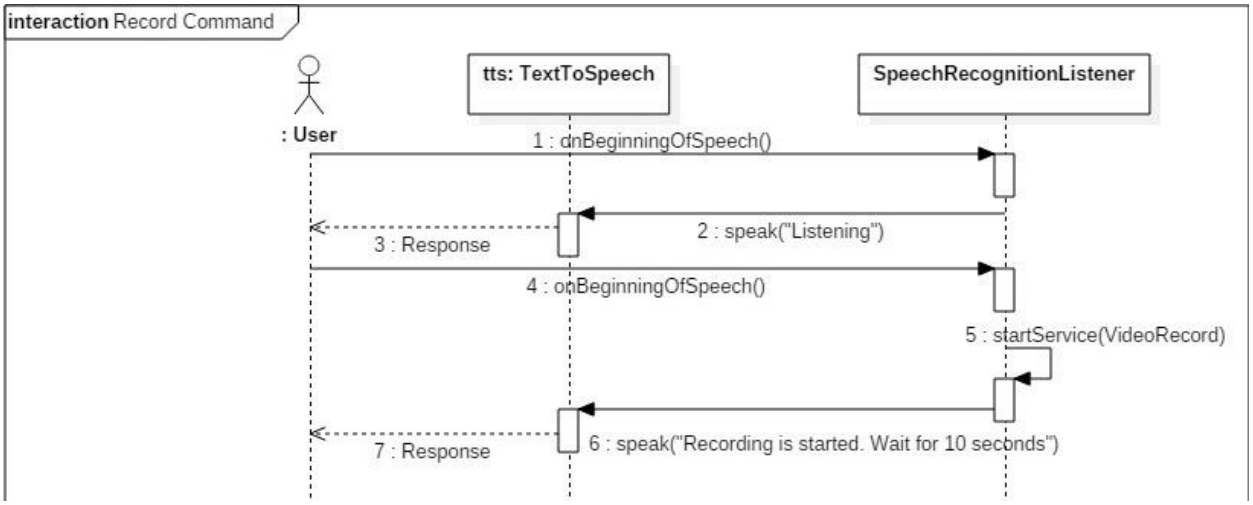

Figure 13: Record command sequence diagram

The figure above shows the record command interaction between the user and the application. Initially, user says the starter command and the service enters the listening state. Then user says the record command. After that, the application starts the VideoRecord service and records video on background service by means of video duration time under Settings Activity. If video duration time is not set default value is 10 seconds.

5.7.10 Crash Command Interaction

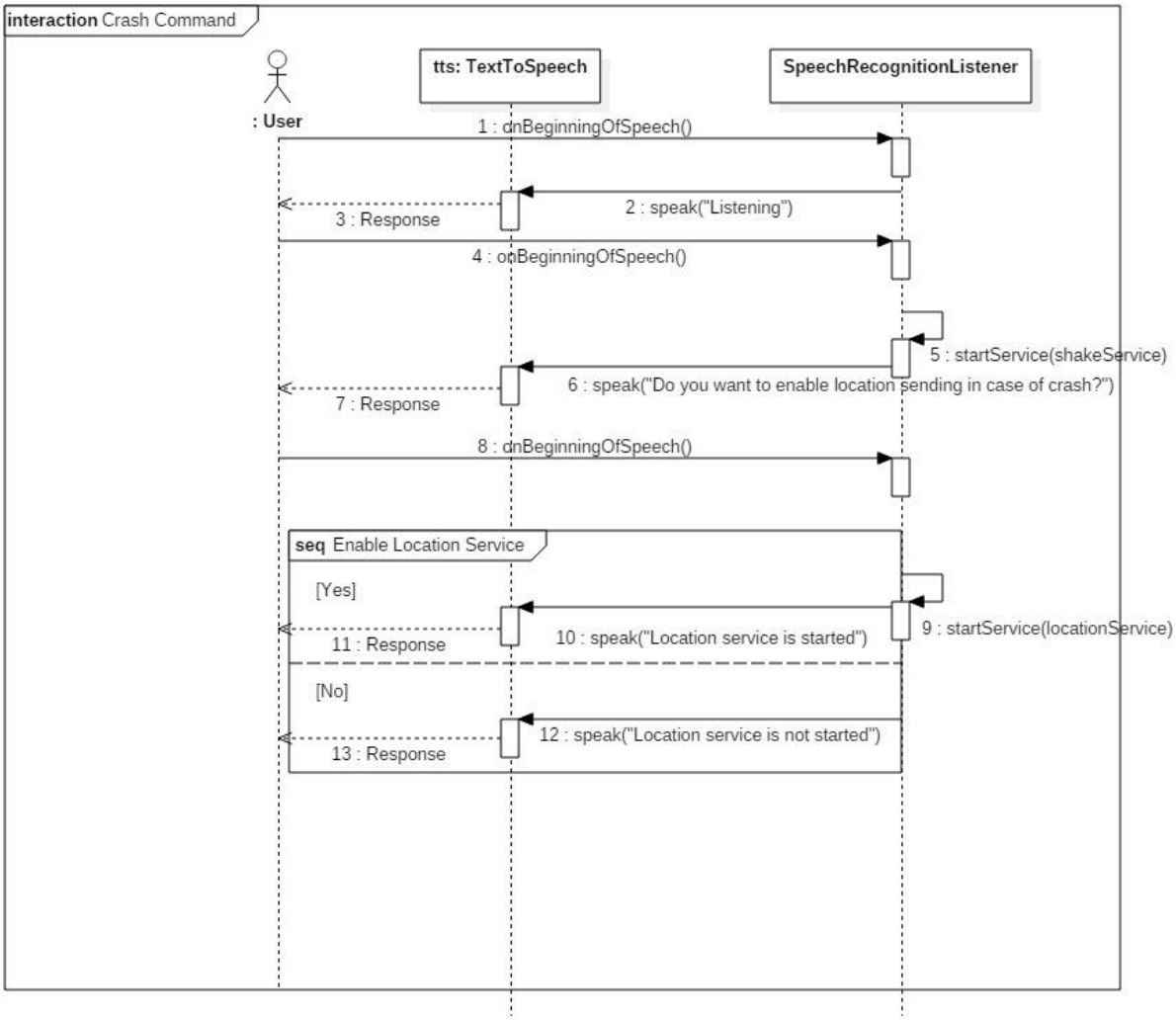

Figure 14: Crash command sequence diagram

The figure above shows the crash command interaction between the user and the application. Initially, user says the starter command and the service enters the listening state. Then user says the crash command. The application starts the shake detection service and asks the user to activate the location sending. If the user says “Yes”, the location service is started. If the user says no, the location service is not started.

5.7.11 Help Command Interaction

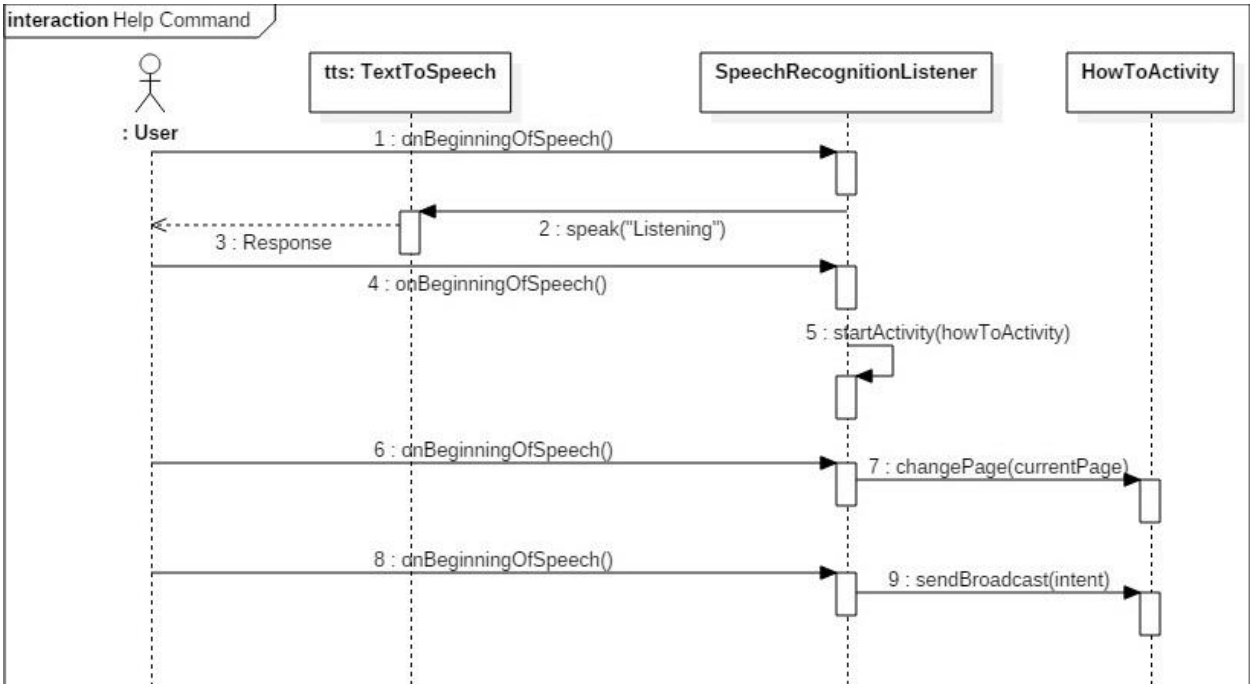

Figure 15: Help command sequence diagram

The figure above shows the help command interaction between the user and the application. Initially, user says the starter command and the service enters the listening state. Then user says the help command and how to activity is started accordingly. If the user says next, next page of the user manual is shown. After the user says exit command, how to activity is closed.

5.8 State dynamics viewpoint

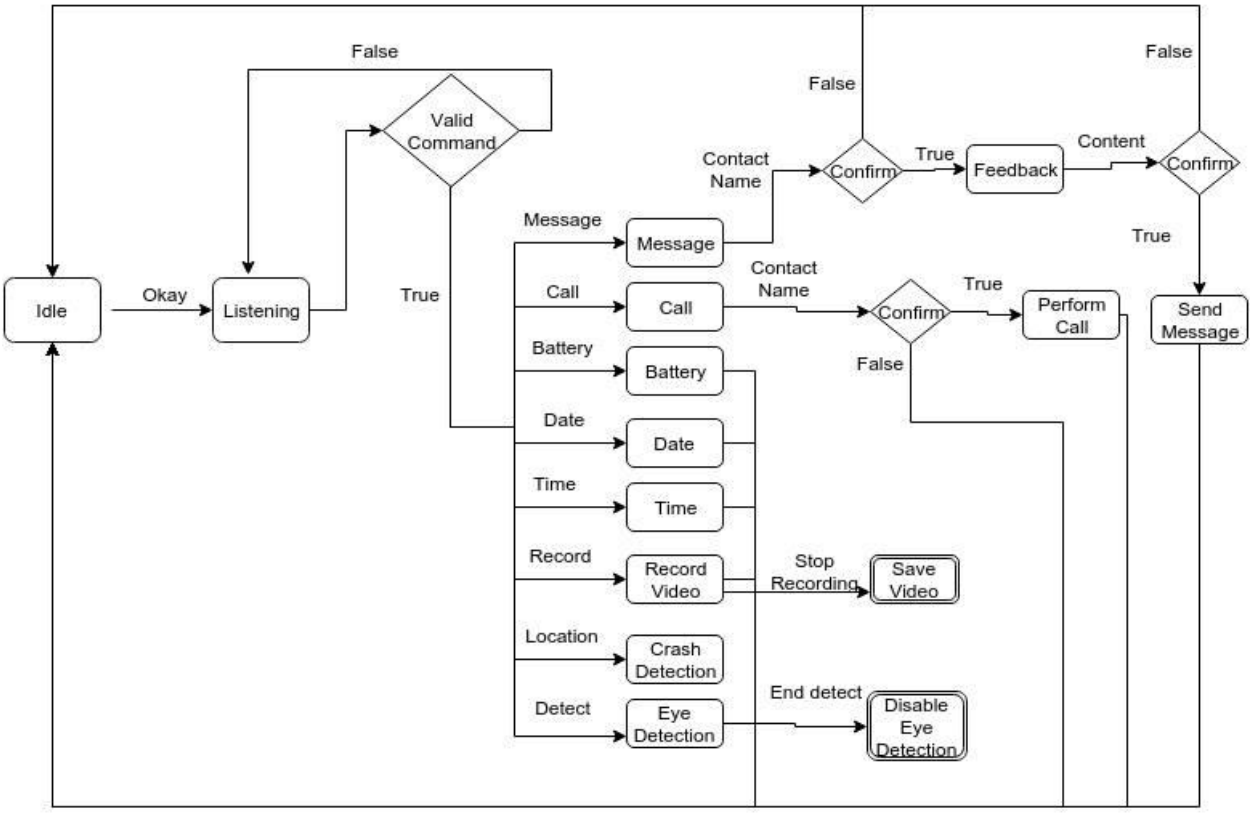

Figure 16: State diagram

5.8.1 Idle

When the application starts, it waits for starter command. After it receives the starter command, it goes the listening state.

5.8.2 Listening

In this state, the application waits for a valid command. If one of the supported command is received, it goes to one of the states to perform a task according to the received command.

5.8.3 Message

If the user says “message” in the listening state, the application enters the message state. When an existing contact name is provided, the application enters the feedback state. If the contact name is not found, the application goes to idle state.

5.8.4 Feedback

In this state, the application waits for the user to say the message content. After the message content received, the application waits for confirmation. If the message is confirmed, it is sent to the contact and application goes to idle state. If the message is not confirmed, the application goes into idle state.

5.8.5 Call

After the user says “call” in the listening state, the application goes to call state. Then the application waits for a contact name. If the contact name is received, the application asks for confirmation. If the user confirms, the application performs call operation and goes to idle state. Otherwise the application directly goes to the idle state.

5.8.6 Date

When the user says “date” in the listening state, the application informs the user about current date and goes to the idle state.

5.8.7 Time

When the user says “time” in the listening state, the application informs the user about current time and goes to the idle state.

5.8.8 Record Video

Whenever user gives the command “record” in the listening state, the application enters the Video Record state. After the recording starts, the application goes to idle state. If the user stops

video recording, video is saved to the hard drive.

5.8.9 Crash Detection

If the user says “crash” in the listening state, the application goes to crash detection state. In this state, the application asks the user whether he/she wants to activate location service. If the user confirms, the location service is activated and the service goes to idle state. If the user does not confirm, the location service does not activated and the application goes to the idle state.

5.8.10 Eye Detection

When the users says “detect” in the listening state, the application enters the Eye Detection state and starts detection process. If the user closes eye detection, the application goes to the idle state.