Smart Driver Assistant
Software Requirements Specifications

SEYMUR MAMMADLI
SHKELQIM MEMOLLA
NAIL IBRAHIMLI
MEHMET KURHAN
Preface

This document contains the software design description for Smart Driver Assistant project. It is prepared in accordance with “ISO-IEC-IEEE 29148-2011 Systems and software engineering — Life cycle processes — Requirements engineering”

All the details which will be needed for building Smart Driver Assistant software project will be provided. Throughout this document use case models, class diagrams, sequence diagrams, object behavior models will be represented including other supporting design information.

During the first section, reader will be informed about the purpose and the scope of this project. Reader will be provided with perspective of this project.

In the second section a more detailed information will be given about the description of this project, the product perspective, product functions, user characteristics, constraints, dependencies and assumptions.

While in the third section, specific and concrete requirements will be shown including external interfaces, functional and non-functional requirements. Reader will be acknowledged with some definitions, abbreviations and acronyms which are used in this document.
# Table of Contents

1. Introduction ........................................................................................................ 4
   1.1 Scope ........................................................................................................... 4
   1.2 Overall Description .................................................................................. 4
      1.2.1 Product Perspective ........................................................................ 5
      1.2.2 Product Functions ............................................................................ 6
      1.2.3 User Characteristics ........................................................................ 7
      1.2.4 Limitations ....................................................................................... 7

2. References ......................................................................................................... 8

3. Specific Requirements ..................................................................................... 9
   3.1 Product Perspective Requirements ......................................................... 9
      3.1.1 User Interfaces .................................................................................. 9
      3.1.2 Software Interfaces .......................................................................... 10
      3.1.3 Hardware Interfaces ....................................................................... 11
      3.1.4 Communications Interfaces ......................................................... 11
      3.1.5 Memory Constraints ....................................................................... 12
      3.1.6 Site Adaptation Requirements ...................................................... 12
   3.2 Functions Requirements .......................................................................... 12
   3.3 Usability Requirements .......................................................................... 22
   3.4 Performance Requirements .................................................................... 22
   3.5 Design Constraints ................................................................................ 23
   3.6 Standard Compliance .......................................................................... 23
   3.7 Software System Attributes ................................................................... 23
      3.7.1 Reliability ......................................................................................... 23
      3.7.2 Availability ....................................................................................... 23
      3.7.3 Security ............................................................................................ 23
      3.7.4 Maintainability ............................................................................... 23
      3.7.5 Portability ......................................................................................... 23
   3.8 Supporting Information ........................................................................... 24
      3.8.1 Configuration Management Tool .................................................... 24

4 Data Model and Description .......................................................................... 24
   4.1 Data Description ..................................................................................... 24
      4.1.1 Data Objects .................................................................................... 24
      4.1.2 Data Dictionary .............................................................................. 25

5. Appendices ..................................................................................................... 27
   5.1 Constraints, Assumptions and Dependencies ......................................... 27
   5.2 Definitions, Acronyms and Abbreviations ............................................. 27
List of Tables and Figures

Figure 1: System Architecture Diagram.........................................................6
Figure 2: Main User Interface.......................................................................9
Figure 3: “How to Screen”..........................................................................10
Figure 4: Preferences Screen......................................................................10
Figure 5: “About us” Screen......................................................................11
Figure 6: Use Case Diagram of SDA..........................................................13
Figure 7: Class Diagram of SDA.................................................................24

Table 1: Use case for Enable and Disable Voice Commands.......................14
Table 2: Use case for Enable and Disable Notifications..............................15
Table 3: Use case for Enable and Disable EDS...........................................15
Table 4: Use case for Get Application Info.................................................16
Table 5: Use case for Handle Phone Call....................................................16
Table 6: Use case for Phone Call.................................................................17
Table 7: Use case for Learn Application.....................................................18
Table 8: Use case for Filter Notification......................................................18
Table 9: Use case for Interacting with EDS...............................................19
Table 10: Use case for Set Personal Info....................................................20
Table 11: Use case for Handle Message.....................................................20
Table 12: Use case for Sending Message....................................................21
Table 13: Use case for Device Status..........................................................22
Table 14: Use case for Shake Level..............................................................23
Table 15: Data Dictionary...........................................................................26
Table 16: List of Definitions, Acronyms and Abbreviations......................27
1. Introduction

This System Requirements Specification aims to mark out and show the design and architecture of Smart Driver Assistant from different aspects. It describes how the system is structured in order to supply the demands mentioned in this document. It is intended to give a hand to the reader to understand the implementation phase.

The main audience for this document is the design and the development team of the Smart Driver Assistant project which is called TETRIS team. Development, testing, and design are all done by the same team.

1.1 Scope

As the phone is used almost by everyone, but no one has a tendency to use it as a tool to save their own lives or the other's lives. So why not take more benefits from scope of features of the phones and extend it for our own welfare. Our project is aimed to be used by everyone who owns an Android phone. Furthermore our application will perform users' command requests, i.e texting or calling. In terms of advantages we can say that the driver will not need to buy anything in order to use this application.

The purpose of this project it to reduce the number of car accidents caused by drivers only because of their phone usage and the cases when they are sleeping. When the phone is used while driving, the driver's concentration is highly diminished, thus it creates a gap for making an error. In other cases when the driver has been driving for hours or is waked up early for job, it creates again a gap for making an error. This error could cause his/her death as well as other people who are driving or passing by. So we aim to take that phone and make it look like a special device which will serve as a guard of the driver and give the driver the opportunity to be fully focused and have a high probability of avoiding any accidents. Moreover crash detection algorithm will be implemented to make emergency calls in case of car accident. The only way of 'communication' between the driver and his/her phone will be his/her voice and the application's voice.

1.2 Overall Description

In this section of SRS document, general description of the factors which influences the system and its requirement is involved. It supplies with diagrams and models which gives a view of how the
system is going to behave, respond and interact with the customer.

1.2.1 Product Perspective

This project is based on voice command feature and image recognition feature of Android operating system. That means it will be an Android application. The drivers mostly use their hands to call somebody or text them. Our application will set the drivers hands free. They will use only their voice to perform these operations i.e. call or text. Another very important feature will be that the phone itself might be placed near the corner of the window or in front of the driver in order to check whether if he/she might be sleeping. If it will be the case so, then the phone will interact with a driver, to wake him/her up. The type of interaction might be an alarm with vibration or a customized message set by the driver itself with maximum volume enabled.

In case of accident or crash detection our product will ask for feedback from driver in case of negative feedback or no feedback emergency call will be made by our product.

This application will execute the commands according to the user request. The following commands will be implemented:
- Call
- Send a message
- Read out a message
- Read out incoming notifications
- Activate safeguard mode

The application itself will interact with the driver for checking his/her conditions of sleeping. It will use the latest and most used messaging applications which are currently on the market, such as Facebook. But it will use also the GSM feature of mobile phones to perform the requests defined above. By GSM features we mean that the commands which are already mentioned, will make use of native call application and messenger application installed on the phone which uses SIM card features.

The overall architecture is shown in Figure 1:
1.2.2 Product Functions

Drivers will be the main users and targets of Smart Driver Assistant project.
Driver will interact with phone using voice commands. Driver will be supplied with the following functionalities without any requirement for physical contact:

- Driver will be able to send the messages through GSM or Internet and will be able to ask to read received messages using Text-To-Speech.
- Driver must be informed about incoming notifications. This notification can be an incoming message from social media.
- Driver must be informed about Phone Status. Application must give information about current battery level and warn the driver to plug the phone to charge.
- Driver will be able to receive incoming call and perform an outgoing call.
- Driver must be alerted or warned when systems detect that he/she is sleeping using Eye Detection algorithm.
- Driver must be able to ask for emergency call when Car crash detected.

Figure 1: System Architecture Diagram
1.2.3 User Characteristics

Software team needs to provide a simple and user friendly interface that is easy to cope with. A midlevel phone user must comprehend the system just with reading simple instruction manual about command list. Command list should not be very complex. An average user should easily understand and memorize commands. A user with small speaking flaws must be tolerated by the system.

1.2.4 Limitations

Since our product will be used lots of services like camera, Speech-To-Text and Text-To-Speech it will consume large amount of power. So in long distances phones must be kept plugged in.

Our product will not be able to perform well in very noisy places. Because of mixed sounds will be hard to process. But it will tolerate the car's own noise.

Performance of driver drowsiness detection will be highly diminished in complete dark places. So our product must not be used in complete dark environment for this purpose.
2. References


Smart Driver Assistant

3. Specific Requirements

3.1 Product Perspective Requirements

3.1.1 User Interfaces

There will be four user interfaces:

- Main user interface
- “How to” user interface
- Preferences screen
- “About us” screen

**Main user interface:** This will be main and most qualified interface. This interface provides user triggering voice commands, notification and eye detection. In other words, voice commands, notifications and eye detection can be turned on and off using this interface.

With the action bar control, user will be able to switch between other UI screens.

![Figure 2: Main User Interface](image)

**How to User Interface:** This interface will provide the user with the information about how to use the application. It includes performable commands that will be processed by voice control.
Preferences screen: In preferences screen, user can enter some private and personal information about him/herself like user name, emergency call number etc.

“About us” screen: This screen gives brief information about the product and the developer team.
3.1.2 Software Interfaces

The System will provide interface for Facebook, and Android API’s intertwined. TinyDB is used because no large database is required. Android API’s will be used for Text-To-Speech, Speech-To-Text, messaging, notification handling receiving and sending calls and for crash detection. OpenCV library will be used for Driver Drowsiness Detection. More detailed specification will be demonstrated in Software Design Description (SDD) document.

3.1.3 Hardware Interfaces

No extra hardware is required except Android phone.

3.1.4 Communication Interfaces

All kind of communication will be established on GSM or Internet which includes: messaging, calling and notification handling.
3.1.5 Memory Constraints

Application should need small memory for storing the data. Since it is a mobile application, it will process the data just using small amount of memory. For example, driver drowsiness detection algorithm will be implemented in a way where it satisfies with at most 10 pictures to process.

3.1.6 Site adaptation requirements

If the product is going to be used in an extreme way/road condition crash detection algorithm should be still working no matter the road surface.

3.2 Function Requirements

The system will perform the following functional requirements

- User can enable and disable voice commands
- User can perform a phone call
- User can handle an incoming phone call
- User can send a message
- User can handle an incoming message
- User can acquire device status
- User can set the shake level
- User can set his/her name and emergency phone number
- User can enable and disable notifications
- User can filter notifications
- User can learn how the application works
- User can get general information about the application
- User can enable and disable Eye Detection System
- User can interact with Eye Detection System
Figure 6: Use case diagram of SDA
## Use Case Name
Enable and Disable Voice Commands

## Use Case ID
SDA-UC01

## Actor
Driver

## Description
This use case describes the event of enabling and disabling the feature of using voice commands.

## Precondition
Device's microphone should be working

### Primary Scenario

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: User clicks “VOICE COMMANDS” button in the main screen</td>
<td>Step 2: System checks state of the button</td>
</tr>
</tbody>
</table>

### Alternative Scenario

- **Step 2:** If the button is OFF system checks whether device's microphone is available. If no problem detected while checking microphone system will set the language to user's local language and turn on Voice Command feature by starting its service. Button state is changed to ON.

- **Step 2:** If the button is ON, its state is changed to OFF. Then system will deactivate its running service.

## Postcondition
Voice Command service is either started or terminated

### Table 1: Use case for Enable and Disable Voice Commands

## Use Case Name
Enable and Disable Notifications

## Use Case ID
SDA-UC02

## Actor
Driver

## Description
This use case describes the event of enabling and disabling the incoming notifications.

## Precondition
Device's API should be greater or equal to 21

### Primary Scenario

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: User clicks “NOTIFICATION” button in the main screen</td>
<td>Step 2: System checks device API value</td>
</tr>
</tbody>
</table>

### Alternative Scenario

- **Step 2:** If the button is OFF system checks whether device's API is greater or equal to 21.
  - If the result is successful then the system will open.
Notification Listener Settings. Then the user will be able to mark and give the SDA permissions to read incoming notifications. The notification service will start to run in background. The button state will be changed to ON
- If the result is unsuccessful then the message “This device does not support notification reading” will appear at the bottom of the screen.

**Step 2:** If the button is ON the system will again open Notification Listener Settings and enable the user to remove SDA from reading the notifications. The button state will be changed to OFF and notification service is deactivated.

| Postcondition | Notification service is either started or terminated |

Table 2: Use case for Enable and Disable Notifications

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Enable and Disable EDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case ID</td>
<td>SDA-UC03</td>
</tr>
<tr>
<td>Actor</td>
<td>Driver</td>
</tr>
<tr>
<td>Description</td>
<td>This use case describes the event of enabling and disabling the Eye Detection System</td>
</tr>
<tr>
<td>Precondition</td>
<td>Device's camera should be working</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Primary Scenario</th>
<th>Actor Action</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Step 1: User clicks “EYE DETECTION” button in the main screen</td>
<td>Step 2: System checks state of the button</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alternative Scenario</th>
<th>Step 2: If the button is OFF system checks whether device's camera is available.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- If no problem detected while checking camera system will open the camera and start EDS activity. The button state will be changed to ON</td>
</tr>
<tr>
<td></td>
<td>- If camera is not available then the message “This device camera is unavailable for EDS” will appear at the bottom of the screen.</td>
</tr>
</tbody>
</table>

|                      | Step 2: If the button is ON, its state is changed to OFF. Then system will release the camera and close EDS activity. |

| Postcondition | EDS is either started or terminated |

Table 3: Use case for Enable and Disable EDS
<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Get Application Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case ID</td>
<td>SDA-UC04</td>
</tr>
<tr>
<td>Actor</td>
<td>Driver</td>
</tr>
<tr>
<td>Description</td>
<td>This use case describes the event of introducing the driver with the general information of the application</td>
</tr>
<tr>
<td>Precondition</td>
<td>None</td>
</tr>
<tr>
<td>Primary Scenario</td>
<td></td>
</tr>
<tr>
<td><strong>Actor Action</strong></td>
<td><strong>System Response</strong></td>
</tr>
<tr>
<td>Step 1: User clicks “About” in the task bar of the main screen</td>
<td>Step 2: System opens the information dialog</td>
</tr>
<tr>
<td>Alternative Scenario</td>
<td>None</td>
</tr>
<tr>
<td>Postcondition</td>
<td>User is presented with the information dialog</td>
</tr>
</tbody>
</table>

**Table 4:** Use case for Get Application Info

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Handle Phone Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case ID</td>
<td>SDA-UC05</td>
</tr>
<tr>
<td>Actor</td>
<td>Driver</td>
</tr>
<tr>
<td>Description</td>
<td>This use case describes the event of handling an incoming phone call</td>
</tr>
<tr>
<td>Precondition</td>
<td>Voice Commands service should be working</td>
</tr>
<tr>
<td>Primary Scenario</td>
<td></td>
</tr>
<tr>
<td><strong>Actor Action</strong></td>
<td><strong>System Response</strong></td>
</tr>
<tr>
<td>Step 3: User gives the command</td>
<td>Step 1: System reads out the name and surname of the caller</td>
</tr>
<tr>
<td></td>
<td>Step 2: System starts Speech-To-Text and waits for the user response</td>
</tr>
<tr>
<td>Alternative Scenario</td>
<td>Step 3: If the command is “Accept” system opens the phone call</td>
</tr>
<tr>
<td></td>
<td>Step 3: If the command is “Reject” system closes the phone call immediately</td>
</tr>
<tr>
<td>Postcondition</td>
<td>User either accepts or rejects the phone call</td>
</tr>
</tbody>
</table>

**Table 5:** Use case for Handle Phone Call
### Use Case: Phone Call

**Use Case Name**: Phone Call  
**Use Case ID**: SDA-UC06  
**Actor**: Driver  
**Description**: This use case describes the actions taken for performing a phone call.

**Precondition**: Voice Commands service should be working.

**Primary Scenario**

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: User starts the call method by saying loudly “call”</td>
<td>Step 2: System asks the user about the callee</td>
</tr>
<tr>
<td>Step 3: User provides the information</td>
<td>Step 4: System processes the information</td>
</tr>
<tr>
<td></td>
<td>Step 5: System performs the phone call</td>
</tr>
</tbody>
</table>

**Alternative Scenario**

**Step 4**: If the information contains a name, system checks that name in phone contacts.
- If the name is unique system moves to Step 5
- If there multiple names, system reads out all of them including their phone's number last 3 digit and ask the user which of them to call. User mention only last 3 digits. If the system correctly receives it, moves to Step 5 otherwise repeats from Step 2

**Step 4**: If the information contains only numbers, system does string processing for errors.
- If no error found, system moves to Step 5
- If error found, system repeats from Step 2

**Postcondition**: System performs the call with the provided information.

---

### Table 6: Use case for Phone Call

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Learn Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case ID</td>
<td>SDA-UC07</td>
</tr>
<tr>
<td>Actor</td>
<td>Driver</td>
</tr>
<tr>
<td>Description</td>
<td>This use case describes the event of introducing the driver with the components, functionalities and usage of the application</td>
</tr>
</tbody>
</table>
### Smart Driver Assistant

**Table 7:** Use case for Learn Application

<table>
<thead>
<tr>
<th>Precondition</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Primary Scenario</strong></td>
<td><strong>Actor Action</strong></td>
</tr>
<tr>
<td></td>
<td>Step 1: User clicks menu button in the main screen</td>
</tr>
<tr>
<td></td>
<td>Step 3: User clicks “How To” menu item</td>
</tr>
<tr>
<td><strong>Alternative Scenario</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Postcondition</strong></td>
<td>User is directed to “How To” activity</td>
</tr>
</tbody>
</table>

**Table 8:** Use case for Filter Notification

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Filter Notification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use Case ID</strong></td>
<td>SDA-UC08</td>
</tr>
<tr>
<td><strong>Actor</strong></td>
<td>Driver</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>This use case describes the event of choosing the desired applications for reading out the incoming notifications</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>Device's API must be greater or equal to 21</td>
</tr>
<tr>
<td><strong>Primary Scenario</strong></td>
<td><strong>Actor Action</strong></td>
</tr>
<tr>
<td></td>
<td>Step 1: User clicks “Settings” button in task bar of the main screen</td>
</tr>
<tr>
<td></td>
<td>Step 3: User marks the desired applications</td>
</tr>
<tr>
<td><strong>Alternative Scenario</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Postcondition</strong></td>
<td>Notification Listener list is updated accordingly</td>
</tr>
</tbody>
</table>

**Table 8:** Use case for Filter Notification
<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Interacting with EDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case ID</td>
<td>SDA-UC09</td>
</tr>
<tr>
<td>Actor</td>
<td>Driver</td>
</tr>
<tr>
<td>Description</td>
<td>This use case describes the actions taken by the system when interacting with user using EDS</td>
</tr>
<tr>
<td>Precondition</td>
<td>EDS service must be working</td>
</tr>
<tr>
<td>Primary Scenario</td>
<td></td>
</tr>
</tbody>
</table>

### Step 1: System detects the face
- Says loudly “Face detected, ready for watching” and waits for user’s response

### Step 2: User when being ready gives the command by saying loudly “ready”

### Step 3: System starts monitoring driver's eye

### Alternative Scenario
- **Step 3:** System detects the eyes are closed and asks the question “username?”
  - If no answer received or the answer does not contain “yes” keyword, then the alarm will ring
  - If the answer is “yes”, no action will be taken

| Postcondition       | None                              |

**Table 9:** Use case for Interacting with EDS

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Set Personal Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case ID</td>
<td>SDA-UC10</td>
</tr>
<tr>
<td>Actor</td>
<td>Driver</td>
</tr>
<tr>
<td>Description</td>
<td>This use case describes the event of setting user's personal information</td>
</tr>
<tr>
<td>Precondition</td>
<td>None</td>
</tr>
<tr>
<td>Primary Scenario</td>
<td></td>
</tr>
</tbody>
</table>

### Step 1: User clicks “Settings” button in task bar of the main screen

### Step 2: System opens the settings activity

### Step 3: User enter personal

19
information in the related fields

Step 4: System saves the data and use the information when necessary.

Alternative Scenario

None

Postcondition

User’s personal information is updated

<table>
<thead>
<tr>
<th>Table 10: Use case for Set Personal Info</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Handle Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case ID</td>
<td>SDA-UC11</td>
</tr>
<tr>
<td>Actor</td>
<td>Driver</td>
</tr>
<tr>
<td>Description</td>
<td>This use case describes the event of handling an incoming message</td>
</tr>
<tr>
<td>Precondition</td>
<td>Voice Commands service should be working</td>
</tr>
<tr>
<td>Primary Scenario</td>
<td>Actor Action</td>
</tr>
<tr>
<td></td>
<td>Step 3: User gives the command</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Alternative Scenario</td>
<td>Step 3: If the command is “Read” system find the message and reads it loudly</td>
</tr>
<tr>
<td></td>
<td>Step 3: If the command is “Ignore” system takes not actions</td>
</tr>
<tr>
<td>Postcondition</td>
<td>User either reads or ignore the message</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 11: Use case for Handle Message</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Send Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case ID</td>
<td>SDA-UC12</td>
</tr>
<tr>
<td>Actor</td>
<td>Driver</td>
</tr>
<tr>
<td>Description</td>
<td>This use case describes the actions taken for sending a message</td>
</tr>
<tr>
<td>Precondition</td>
<td>Voice Commands service should be working</td>
</tr>
<tr>
<td>Primary Scenario</td>
<td>Actor Action</td>
</tr>
<tr>
<td></td>
<td>Step 1: User starts the message method by saying loudly</td>
</tr>
</tbody>
</table>
“message”

Step 3: User provides the information

Step 6: User composes the content.

Step 8: User gives response

Step 2: System asks the user about the receiver

Step 4: System processes the information

Step 5: System asks user about the message content

Step 7: System reads out loudly the content and ask user for confirmation

Step 8: System processes the response

Alternative Scenario

Step 8: System will decides on response

- If the response is “send”, system will send the message
- If the response is “revise”, system will repeat from Step 5
- If the response is “discard”, system will terminate the command

Postcondition

System either sends or discards the message

Table 12: Use case for Send Message

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Device Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case ID</td>
<td>SDA-UC13</td>
</tr>
<tr>
<td>Actor</td>
<td>Driver</td>
</tr>
<tr>
<td>Description</td>
<td>This use case describes the event of requiring device's status</td>
</tr>
<tr>
<td>Precondition</td>
<td>Voice Commands service should be working</td>
</tr>
<tr>
<td>Primary Scenario</td>
<td>Actor Action</td>
</tr>
<tr>
<td></td>
<td>Step 1: User starts the command by saying out loudly “battery”</td>
</tr>
<tr>
<td>Alternative Scenario</td>
<td>None</td>
</tr>
</tbody>
</table>

21
Smart Driver Assistant

| Postcondition | User is informed about the device's status |

**Table 13**: Use case for Device Status

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Shake Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case ID</td>
<td>SDA-UC14</td>
</tr>
<tr>
<td>Actor</td>
<td>Driver</td>
</tr>
<tr>
<td>Description</td>
<td>This use case describes the event of setting shake level sensor's value</td>
</tr>
<tr>
<td>Precondition</td>
<td>None</td>
</tr>
<tr>
<td>Primary Scenario</td>
<td><strong>Actor Action</strong></td>
</tr>
<tr>
<td></td>
<td>Step 1: User swipes the seek bar from left to right or from right to left</td>
</tr>
<tr>
<td>Alternative Scenario</td>
<td>None</td>
</tr>
<tr>
<td>Postcondition</td>
<td>Shake level in the shake service is updated</td>
</tr>
</tbody>
</table>

**Table 14**: Use case for Shake Level

### 3.3 Usability requirements

The usability requirements of Smart Driver Assistant are the following:

- The system must provide a simple and user friendly GUI
- System must be robust to extreme road conditions.
- System must be robust to user accent.
- The number of physical interactions should be minimum

### 3.4 Performance requirements

The performance requirements of Smart Driver Assistant are the following:

- It should process voice command in less than 2 seconds.
- It should process the Eye Detection algorithm less than 0.3 second
3.5 Design Constraints

The design constraints of the Smart Driver Assistant are listed as below:

- The program will be implemented using Android studio.
- Database side will be implemented using TinyDB
- Minimum Android API level must be higher than 15.
- User private data and actions must be kept private.

3.6 Standard Compliance

The system will wait for input voice command after being triggered by the keyword “Tetris”.

3.7 Software System Attributes

The software system attributes of Smart Driver Assistant can be listed below:

3.7.1 Reliability

- Smart Driver Assistant shall be run properly at every time needed
- Failure intensity should not be acceptable

3.7.2 Availability

- Smart Driver Assistant shall be implemented so that application crashes will be minimum
- Recovery of the whole system should take minimum time

3.7.3 Security

- User also need to activate notification access permission on device manually
- User's personal info shall not be accessed or reached by anyone except that person who can learn user location in case of emergency.

3.7.4 Maintainability

- System shall responds to the real time and physical interactions
- System shall responds to user commands properly
- Codes shall be supported by some techniques like related implementation comments, naming conventions and coding standards for increasing readability for future developments

3.7.5 Portability
Smart Driver Assistant

- The Application shall be accessed also by different electronic devices using Android Operation System
- The Application shall be working properly on different versions of Android Operation System

3.8 Supporting Information

3.8.1 Configuration Management Tool
- The source code developed for this system shall be maintained in CMT

4. Data Model and Description

4.1 Data Description

4.1.1 Data Objects
Class diagram of Smart Driver Assistant is shown below:

[Diagram of Smart Driver Assistant]

Figure 7: Class Diagram of SDA

4.1.2 Data Dictionary

| **Main Activity** | This class is the starting point of our project. It starts all the services. |

24
## 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.

g etAllNotifications(): 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 15: Data Dictionary

5. Appendices

5.1 Constraints, Assumptions and Dependencies
• It is assumed that there will be no broken bicycles which might be inappropriately shown in the User Interface.
• There must be at least one maintainer during the working time in each station.
• There must be a clone copy saved in case of System and Database crash.

5.2 Definitions, Acronyms and Abbreviations

The definitions of the terms, which are used in this SRS document, are shown below:

<table>
<thead>
<tr>
<th>TERMS</th>
<th>DEFINITIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTS</td>
<td>Text To Speech</td>
</tr>
<tr>
<td>STT</td>
<td>Speech To Text</td>
</tr>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>SDA</td>
<td>Smart Driver Assistant</td>
</tr>
<tr>
<td>EDS</td>
<td>Eye Detection System</td>
</tr>
<tr>
<td>CMT</td>
<td>Configuration Management Tool</td>
</tr>
</tbody>
</table>

Table 16: List of Definitions, Acronyms and Abbreviations