**Smart Vocal List**

By:

Aharon Garada

Ronen Rossin

Alexander Manevitch

Supervisor:

Eliav Menachi

Git:

Ling to the git repository

**Table of Contents**

[**1.**](#_gjdgxs) **PROJECT DESCRIPTION 2**

[**2.**](#_30j0zll) **RELATED WORK 2**

[**3.**](#_1fob9te) **FUNCTIONAL DESCRIPTION / REQUIREMENTS 2**

[**4.**](#_3znysh7) **ARCHITECTURE 3**

[4.1.](#_2et92p0) Each Module description 3

[**5.**](#_tyjcwt) **WORK PLAN 3**

[**6.**](#_3dy6vkm) **CLIENT SIDE 3**

[6.1.](#_1t3h5sf) Usage Illustration 3

[6.2.](#_4d34og8) Mockup 3

[**7.**](#_2s8eyo1) **SERVER SIDE 3**

[7.1.](#_17dp8vu) API 4

# Project description

Smart Vocal List system, suit various applications, it enable to manage lists, read the lists and accept voice feedback from user.

No more paper lists, rather than reporting visually via paper or computer monitors, this system relies on voice instructions through mobile devices such smartphone.

By freeing the user’s hands & eyes, user’s attention and the need of remembering ordered lists, Smart Vocal List ensures focusing the mission and verifying the mission completion as predefined.

The voice dictation reduces user visual workload resulting in better awareness and enhanced productivity.

Smart Vocal List allows users to create their checklist to their specific needs or mission profile.

# Related Work

This section gives a brief description of the related work which is relevant to your project

# Functional Description / Requirements

Smart Vocal List include management web application, and client application which run on mobile devices.

## General

## Users & Roles

The system manage two type of users: Regular and Administrator

Regular user:

Permitted to download lists that are shared with him, report a results and share lists with other users.

Administrator user:

Permitted all Regular user permissions and in addition has the permission to view lists of other users, manage users and its permissions.

## Server side application

Should be implemented as Web application, shall enable administrator/regular user to define these entities.

* User first time registration, for start use the system.
* User can invite other people for registering and start use the system.
* User can change its own account profile data
* Define Categories
* Define a checklists
* Define checklist items - each checklist item should contain an option for reporting a result by user.
* Each item result type should be defined as one of the following options:

Numeric, Alpha-numeric, Date, Time, Boolean (Yes/No, Pass/Failed etc.) and Image (user may be required to take a picture as a result of checklist item).

* Privacy definition – user can set a checklist as a private/public/share with specific users.
* User can see its private entities, as well as public entities.
* User may share its own individual entity with other users.
* User can select a checklist, create a new instance of it and start reporting a results accordingly.
* Checklist instance displayed on the mobile screen, user can mark the checklist as completed or deferred.
* User can report a result for each item of checklist, manually.
* User can view its own checklist results (current and history)
* User can share his checklist results, with all users or with individual user
* User which belong to Administrators Role can view as well as other user’s results (improvement).
* Users management – set role for users (improvement), block users from reporting, update/delete users account.

## Client side application

Should be implemented as Mobile App, Shall enable:

* User first time registration, for start use the system.
* User can invite other people for registering and start use the system.
* Define a checklists
* Define checklist items - each checklist item should contain an option for reporting a result by user.
* Each item result type should be defined as one of the following options:

Numeric, Alpha-numeric, Date, Time, Boolean (Yes/No, Pass/Failed etc.) and Image (user may be required to take a picture as a result of checklist item).

* Privacy definition – user can set a checklist as a private/public/share with specific users.
* User can see its private entities, as well as public entities.
* User may share its own individual entity with other users.
* User can select a checklist, create a new instance of it and start reporting a results accordingly.
* Checklist instance displayed on the mobile screen, user can mark the checklist as completed or deferred.
* User can request the system to read the selected checklist instance.
* User can report a result for each item of checklist, manually or vocally (when possible – e.g. Image results etc.).
* User can view its own checklist results (current and history)
* User can share his checklist results, with all users or with individual user
* User which belong to administrators Role can view as well as other user’s results (improvement).
* User can mark the checklist as completed or deferred.

# Architecture

This section will present the overall architecture of the project in terms of the components that will construct the system: server side, client side, DB, external resources. It should be graphically presented with a short description of each main module of the system.

Project with deep learning and algorithms that are based on large data must provide the source of the data in this section

## *Each Module description*

In this sub section you will describe the responsibility of each module… in case the module is a combination of several sub modules you should drill down and describe each of this sub modules as well.

# Work plan

# Client side

## *Usage Illustration*

1. Regular usage :

Logs in

Connects to the server and downloads new lists that are uploaded for him

Uses the list , all goes right so hi finishes it successfully

2) Usage with a non fatal error :

Logs in

Connects to the server and downloads new lists that are uploaded for him

Uses the list , one test fails but this is not a fatal error so he continues and finished

3) Usage with a fatal error

Logs in

Connects to the server and downloads new lists that are uploaded for him

Uses the list , one test fails , the error is fatal the whole list fails

## *Mockup*

**Android application on smartphone:**

Login screen:



Main screen:



**Web application in any browser:**

Main screen:

# Server Side

The server side which is developed in C# (Specific technology TBD) will manage the users and the user permissions will be able to deliver lists that are created by groups of users to the users they choose to share the list with.

The server will also provide statistics for Administrator and allow them to control the permissions of users.

## zx

## *API*

This is the API that the server present to the rest of the world.