

# Machine Vision application interface documentation

V ALPHA 1.0

LINARDS LIEPENIEKS

# Introduction

## Overview

MachineVision Interface is created to allow user speech and text input to be delivered to a webserver for further analysis and delivery to machines for execution.

The primary functionality of this interface is to:

- a) Connect to a remote webserver and transmit user audio and text input
- b) Manage real-time connections between users of the interface and connected machines

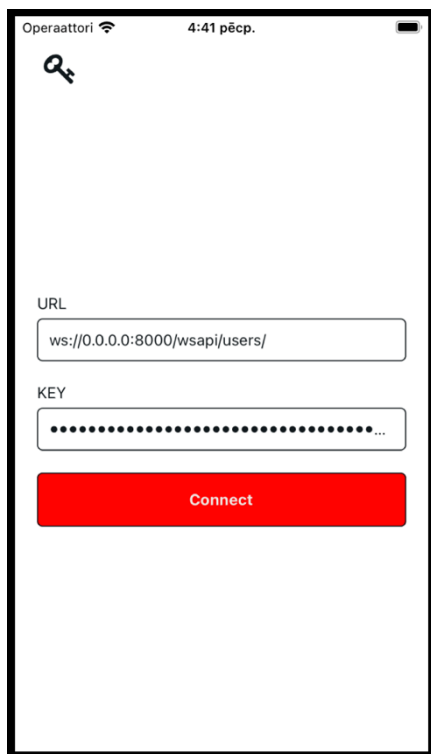
The current MachineVision interface is meant to be used in conjunction with the NLP Server Django application developed during this project.

The interface is built on react-native (0.75.2) with Expo framework (51.0.31)

## Functionality

### Screens

#### Connect



#### Credentials Button:

**Description:** Located in the top, left corner marked with a key symbol is the Credentials screen navigation button.

**Functionality:** Navigates to the Credentials screen.

#### URL Input:

**Description:** Input for the URL of the NLP Server.

**Functionality:** Upon connection attempt retrieves the URL and attempts to connect to the described endpoint.

**Notes:** The application *DOES NOT* append or prepend any information (e.g. protocol or the endpoint) so the URL should be entered in its entirety.

#### Key Input:

**Description:** Input for the API Key of the user.

**Functionality:** Visually hides the API key. Upon connection attempt key is retrieved and sent to the server for validation.

**Notes:** If using with an unmodified instance of the

NLP Server only user API key should be entered.

#### ConnectButton:

**Description:** Button for attempting a connection with the server. Button has three states – Connect, Connecting, Connected.

**Functionality:** On press button URL and Key is retrieved and a connection to the server is attempted.

Button should change state to connecting while connection is attempted.

Button should change state to connected if connection is successful as well as navigate to the Record screen and store the credentials in local storage (if these credentials are not present).

In the case of empty input fields an alert should be shown instructing to fill the fields.

In the case of an unsuccessful connection an alert should be shown informing the user of failure.

### Proposed improvements:

- As URL is not prepended or appended any URL it is possible to connect to an endpoint not meant for working with this application – there should be a more strict connection mechanism.
- Returned error messages upon failed connection should be more descriptive – requires further testing.

## Credentials



### Credential List

**Description:** A FlatList element which renders all stored credentials.

**Functionality:** Renders Credential objects present on local storage.

### Credential

**Description:** A pressable element which displays the URL, first 5 symbols of the API key and has a Remove button.

**Functionality:** On press throws an alert with confirmation whether to connect using these credentials – if response is “ok” connects to the server using these credentials.

### CredentialRemove Button

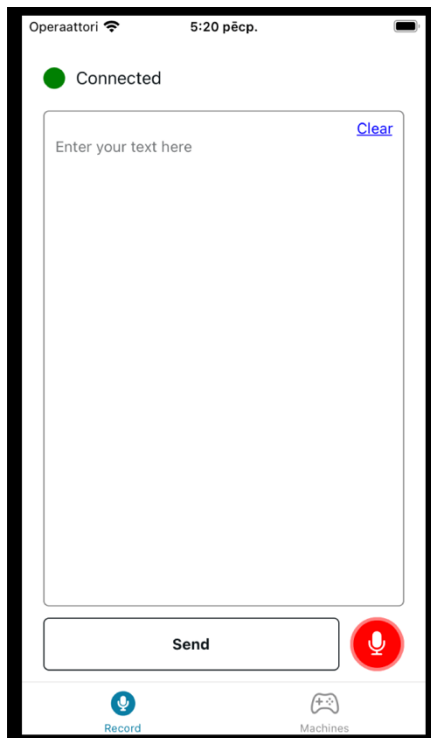
**Description:** Aligned with the bottom of Credential object, generated with each Credential object a pressable button.

**Functionality:** Removes the chosen credential from local storage.

### Known Bugs:

- Sometimes when credential connection is pressed it does not attempt connection, the Credential object needs to be pressed twice to initiate connection – most likely this is connected to whether the input fields are filled or not. This should be fixed in the future, however currently it is not of high priority so is left unfixed.

## Record



### ConnectionStatus

**Description:** Located on top left of the screen, serves as an indicator whether server is available. Has the functionality to disconnect. Has 3 states, connected, connecting, disconnected.

**Functionality:**

- Monitors the ws connection - upon lost connection changes state to disconnected.
- On press terminates current ws connection

### TextAreaRecordingInput

**Description:** Textarea input will fit the height of the screen. Upon vertical text overflow enables scrolling.

**Functionality:** Allows user text input, receives transcribed text upon transcription response. Upon Recording and transcription locks and displays progress of transcription response.

### ClearButton

**Description:** Located top right of

TextAreaRecordingInput.

**Functionality:** On press clears all text inside TextAreaRecordingInput.

### MessageButton

**Description:** Located bottom of screen. Delivers user input to the server.

**Functionality:** On press attempts to send a message containing TextAreaInputText to the server.

### RecordButton

**Description:** Located bottom of screen. Records audio and send for transcription.

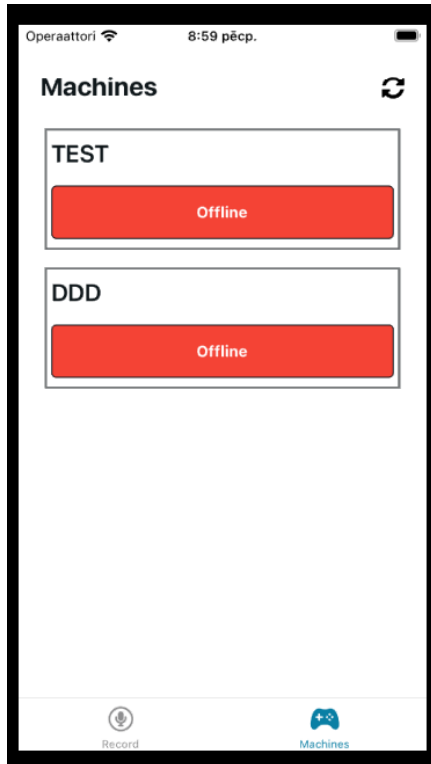
**Functionality:** While button is held it records audio which is stored on local storage, upon release sends audio file to server for transcription. After delivering audio the file is deleted.

## Proposed improvements

- ConnectionStatus component should have some type of reconnection mechanism which attempts to reconnect to the server upon a broken connection.
- RecordButton functionality should be optimized as it takes a large amount of time to retrieve transcribed text from the server. This can be done when architecture of the server is decided and whether transcription service will be hosted locally, or a cloud API will be used.
- SendMessage functionality should be optimized as it is slow, although this has more to do with the server implementation – one possible solution might be a task queue for the incoming messages.

- TextAreaRecordingInput should emphasize the keywords which NLPServer recognizes – objects, actions and attributes should be colored/underlined.

## Machines



### RefreshButton

**Description:** Pressable, located top right corner of screen. Marked with cyclic arrows.

**Functionality:** On press sends a message to the API to reload the machine list.

### MachineList

**Description:** Flatlist, which displays a list of all machines available to user.

**Functionality:** Upon every render (reload of the page or navigation from other pages) reloads the machine list by retrieving machines from the API

### Machine

**Description:** Machine component which displays the available machines name and has a connection button.

**Functionality:** Connection button has 3 states:

- Connect – machine is connected to the api and available: On press attempts to connect to machine
- Offline – machine is not connected to the api and

unavailable: On press ignored

- Busy – machine is connected to the Api, but another user is connected – On press ignored
- Disconnect – machine is connected to the Api and client user is connected to it: On press attempts to disconnect

If connection result is unsuccessful user is notified with an alert.

## Proposed improvements

- Error messages should be more fleshed out as currently they just display “error” instead of the whole message, but they should not simply show the whole response query as it is unintuitive – some type of descriptions should be added to response codes – further testing needed.
- Currently machines are reloaded every time the page is reloaded or a machine connection sends a message this should be optimized – further testing needed.

## Contexts

In React and React-Native contexts are essentially services to deliver global variables. In this project there are 2 contexts

- Credential context – manages stored and current authentication credentials

- Connectivity context – manages current ws connection

This section is particularly important for Connectivity context as it describes the functionality of the API connection system, and which messages are being sent and received. If a developer wishes to extend the functionality of the system, they should become closely familiar with its current capabilities.

## Connectivity context

Connectivity Context is responsible for handling all outgoing and incoming messages relating to the API

### Interfaces

- Message – an interface describing incoming and outgoing messages
  - Type (string) – determines type of message
  - Message (string) – determines the contents of message
- Machine – an interface describing a machine to which user may attempt connection
  - Id (number) – id of machine used for database operations
  - Name (string) – a human readable identifier
  - State (string) – current state of the machine
    - Offline
    - Connect
    - Busy
    - Disconnect
  - Key (string) – redis key used for managing real-time connections

### Variables and states

- **url:** The WebSocket server URL
- **key:** The authentication key for the WebSocket connection
- **connectionStatus:** Represents the current connection state to the api ("disconnected", "connecting", or "connected") *NOT INDIVIDUAL MACHINE CONNECTIONS*
- **lastTranscribedMessage:** Stores the most recent transcribed message received from the server – used for receiving transcribed text from audio
- **machines:** An array of connected machines, each with properties like id, name, state, and key

### Messages

#### Outgoing messages:

- **sendAnalyzeMessage:** Sends a text message to the server for analysis
- **sendBinaryMessage:** Transmits binary data (e.g., audio files) to the server for transcription
- **sendReloadMachinesMessage:** Requests the server to reload the list of available machines
- **sendConnectToMachineMessage:** Initiates a connection to a specific machine

#### Incoming messages:

- **auth\_response:** Confirms successful authentication with the server
- **transcribe\_response:** Contains transcription results from the server

- ***update\_machines***: Updates the list of available machines
- ***machine\_connection\_status***: Provides status updates for machine connections

#### Connection management:

- ***handleConnect***: Establishes a WebSocket connection with the server
- ***handleDisconnect***: Closes the WebSocket connection and resets related states

### Credential context

Credential context is responsible for managing current connection credentials and all stored credentials.

#### Interfaces

- Credential – Represents a stored set of connection credentials.
  - Id (string) - Unique identifier for the credential
  - url (string) - WebSocket server URL
  - key (string) - Authentication key for the WebSocket connection

#### Variables and states

- ***storedCredentials***: Array of Credential objects representing saved connection information
- ***STORAGE\_KEY***: Constant string used as the key for storing credentials in SecureStore

### Closing remarks

Currently the interface is capable of establishing a connection and transmitting data to the server per requirements of this project, however serious improvements must be made in the future mainly concerning user experience and security, however these improvements can only be made when the final server architecture has been decided which heavily relies on the specificity of work environment as well as speed and flexibility on what networks the service would be hosted.