

VERSION 1.0
AUGUST 15, 2017

TIZEN VOICE RECOGNITION APPLICATION

SUBMITTED BY: SAPTARSHI CHAKRABORTY
GRADUATE STUDENT
UNIVERSITY OF FLORIDA




TABLE OF CONTENTS

| | |
|---|----|
| 1. PROJECT DESCRIPTION..... | 2 |
| 2. SOFTWARE REQUIREMENTS..... | 2 |
| 2.1. User Interface..... | 2 |
| 2.2. Backend API's | 2 |
| 2.3. Build..... | 2 |
| 2.4. Developer Tools | 2 |
| 2.5. Operating System..... | 2 |
| 3. PROJECT SCOPE | 3 |
| 4. BUILD & RUN INSTRUCTIONS | 3 |
| 4.1. Server-side | 3 |
| 4.2. Client Side..... | 4 |
| 5. PROJECT ARCHITECTURE..... | 4 |
| 5.1. Client-side Architecture (VR app) | 4 |
| 5.1.1. Developer Setup and configurations | 5 |
| 5.2. Server-side architecture | 5 |
| 5.2.1. Developer Setup and configurations | 5 |
| 5.2.2. REST API | 6 |
| 5.3. Database architecture..... | 6 |
| 6. Different Activities..... | 7 |
| 6.1. Audio File Transfer activity | 7 |
| Audio File Transfer activity | 8 |
| 7. TEST CASES | 9 |
| 8. Project status | 9 |
| 9. OPEN ISSUES | 10 |
| 10. FUTURE SCOPE | 10 |
| REFERENCE..... | 10 |

1. PROJECT DESCRIPTION

In this project, we are developing an application to record user voice data. This application will be installed on Samsung Smartwatch running Tizen operating system.

This project has three main aspects

- Record the speech from the user and store it in an audio file in Tizen file system.
- Send the audio file to the server
- Use machine learning techniques to extract the information from the audio file

We have designed a simple user interface, using which the user can record the voice data and the RESTful APIs in the backend send this file to the server. User can also listen the latest recorded audio file from the smartwatch.

2. SOFTWARE REQUIREMENTS

Below are the software's used for this project and are required to successfully run it.

2.1. USER INTERFACE

- HTML 5
- JavaScript
- jQuery
- AJAX

2.2. BACKEND API'S

- Java (JRE 1.8 & JDK 8)
- Jersey
- Tizen SDK

2.3. BUILD

- Maven

2.4. DEVELOPER TOOLS

- Tizen studio
- Postman REST client
- Apache Tomcat server

2.5. OPERATING SYSTEM

- Windows 10
- Tizen 2.0 – Emulator and Samsung Gear smartwatch
- Tizen 3.0 - Emulator

3. PROJECT SCOPE

- A simple UI which can perform 4 operations
 - Start recording the user voice input and stop after 5 seconds
 - Play the recorded audio
 - Stop the audio recording
 - Save the recorded audio file
- Backend service in Tizen performs the following functions
 - Store the recorded audio file in the Tizen file system
 - Send the audio file to the server
- The server-side services perform the following functionalities –
 - Save the binary stream as a File object in the server's temp directory.
- The machine learning libraries convert the audio stream into readable text.

4. BUILD & RUN INSTRUCTIONS

4.1. SERVER-SIDE

Steps to build and run the server-side project -

1. Unzip the project file.
2. Start the Apache Tomcat server.
3. Open a command prompt window and move to the project folder.
4. Build using the command “**mvn clean install**” (The project uses Maven as a build tool to generate the **.war** file)
5. Copy the .war file generated in the target folder of the project directory to Tomcat webapps folder and paste it, and start the server.
6. The project should be successfully deployed to the server and now open a browser and enter the URL <http://localhost:8080/tizenserver/file/hello>. This should show a ‘**Tizen server is running**’ message in the browser.
7. From next time, just start the tomcat server and the above url should be accessible. No need to perform Step 1-5.

4.2. CLIENT SIDE

1. Import the project in Tizen studio and check whether all dependencies are resolved or not.
2. Open an emulator or connect a smartwatch with the IDE and install the application.
3. If application installation fails, then create a certificate from 'Certificate Manager' and upload the certificate in the device so that it can access the required privileges and features. (For more on this, see the architecture section).

5. PROJECT ARCHITECTURE

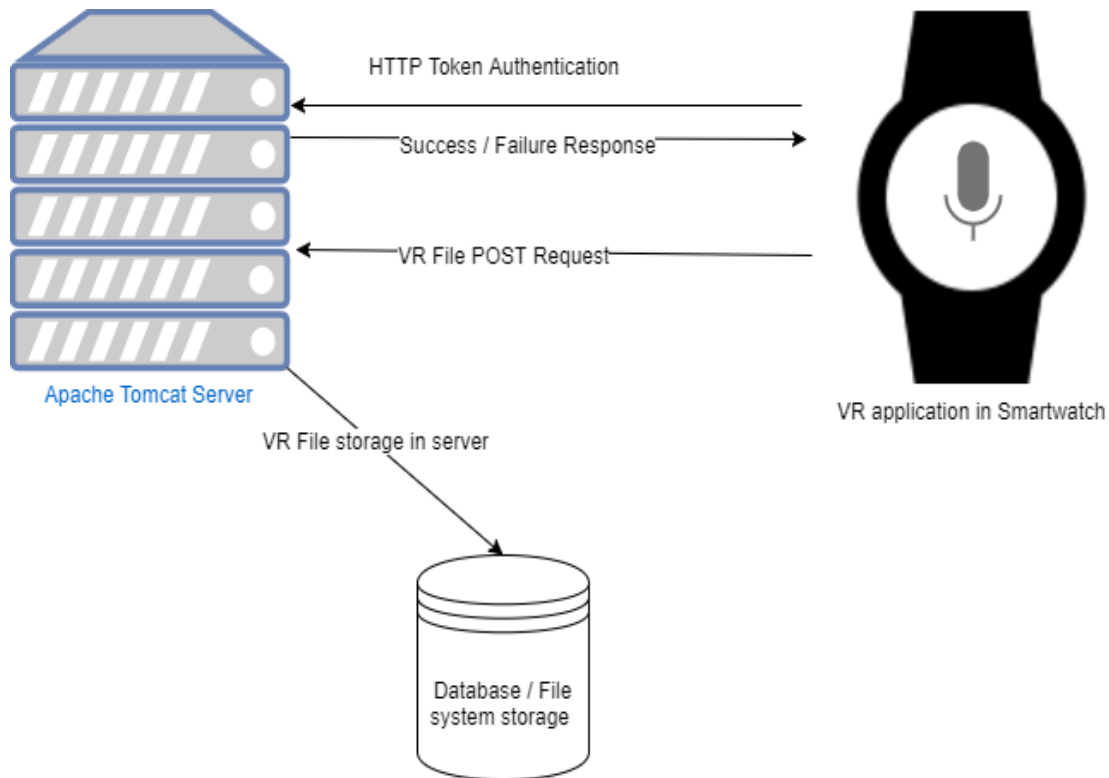


Fig. 1: Architecture and data flow diagram

The architecture can be divided into three modules-

5.1. CLIENT-SIDE ARCHITECTURE (VR APP)

The client-side architecture is deployed in the smart watch, where the voice recording application is installed. This application records audio data and stores it in the watch's file system as an **'.amr'** file. The maximum duration of the audio message is 5 seconds.

5.1.1. DEVELOPER SETUP AND CONFIGURATIONS

- This application accesses the file system to read and write the audio file. So, appropriate permission for read and write should be added in the *privileges*.

```
<tizen:privilege name="http://tizen.org/privilege/filesystem.read"/>
```

- To access the microphone API, we need to set the following two privileges -

```
<tizen:privilege name="http://tizen.org/privilege/mediacapture"/>
```

```
<tizen:privilege name="http://tizen.org/privilege/camera"/>
```

- As the application uses internet connection, so it requires the following privilege to communicate with the external servers

```
<feature name="http://tizen.org/feature/network.internet"/>
```

```
<tizen:privilege name="http://tizen.org/privilege/internet"/>
```

```
<tizen:privilege name="http://tizen.org/privilege/download"/>
```

- The application only sends the recorded audio files, when the smartwatch is connected to WIFI connection. This helps in consuming less battery life cycle as well as sending data over WIFI is more reliable. So, the following privilege is needed to send data over WIFI.

```
<feature name="http://tizen.org/feature/network.wifi"/>
```

5.2. SERVER-SIDE ARCHITECTURE

This architecture is deployed in Apache Tomcat server. The server-side code uses JAVA RESTful APIs to accept the HTTP requests from the smartwatch. For every request from the smartwatch, the server scans the HTTP header to get the authorization header and validates the token present in it. After successful authentication, the server accepts the file stream object, which contains the recorded audio file data. This audio data is then converted to text file using machine learning libraries.

5.2.1. DEVELOPER SETUP AND CONFIGURATIONS

- Install Maven, or Ant, or Gradle as a build tool to generate the final executable package. In our project, I have used Apache Maven to generate the **.war** file.
- Set up the path and environment variables according to the OS type.
- Set up the Tomcat server and check if the port is open or not. Generally, tomcat uses port **8080** to communicate with the client. But it can be configured to communicate using other open ports.

- As per design specification, we are storing both the audio and text file in a temporary directory in the server, and the file path will be stored in the database. So, the directory should be accessible from the server.
- If the server URL will be accessed over internet, then configure the router so that every request for the corresponding port will be redirected to your laptop.

5.2.2. REST API

This API handles the post request from the smartwatch which sends the file to the server. This request contains a **BLOB** object where the filestream is embedded.

For every request, server scans the HTTP header for this particular header field. The complete request for the POST request is –

http:<server IP address>:<port no>/tizenserver/file/send

If the request is processed successfully, server returns HTTP status code **200 (OK)** to the client. If server fails to authenticate the request, then HTTP status code **403 (FORBIDDEN)** is send to the client. For any other failure, such as bad file content, or unsupported media type, HTTP status code **400 (BAD_REQUEST)** is sent to the client along with proper error messages.

5.3. DATABASE ARCHITECTURE

The application uses a relational database to store the processed text data as well as the audio data along with the timestamp. The database structure is defined as below:

| Tizenserver | |
|---------------|---|
| Attribute | Description |
| id | auto generated number used as primary key |
| audiofilepath | absolute path where the audio file is stored in the server file system |
| textfilepath | absolute path in the server where the text content of the recorded voice file is stored |
| timestamp | timestamp for logging and debugging purpose |

The database is hosted in the same machine, where the server is running. This project deals with structured data so we are using relational database to persist the information.

6. DIFFERENT ACTIVITIES

6.1. AUDIO FILE TRANSFER ACTIVITY

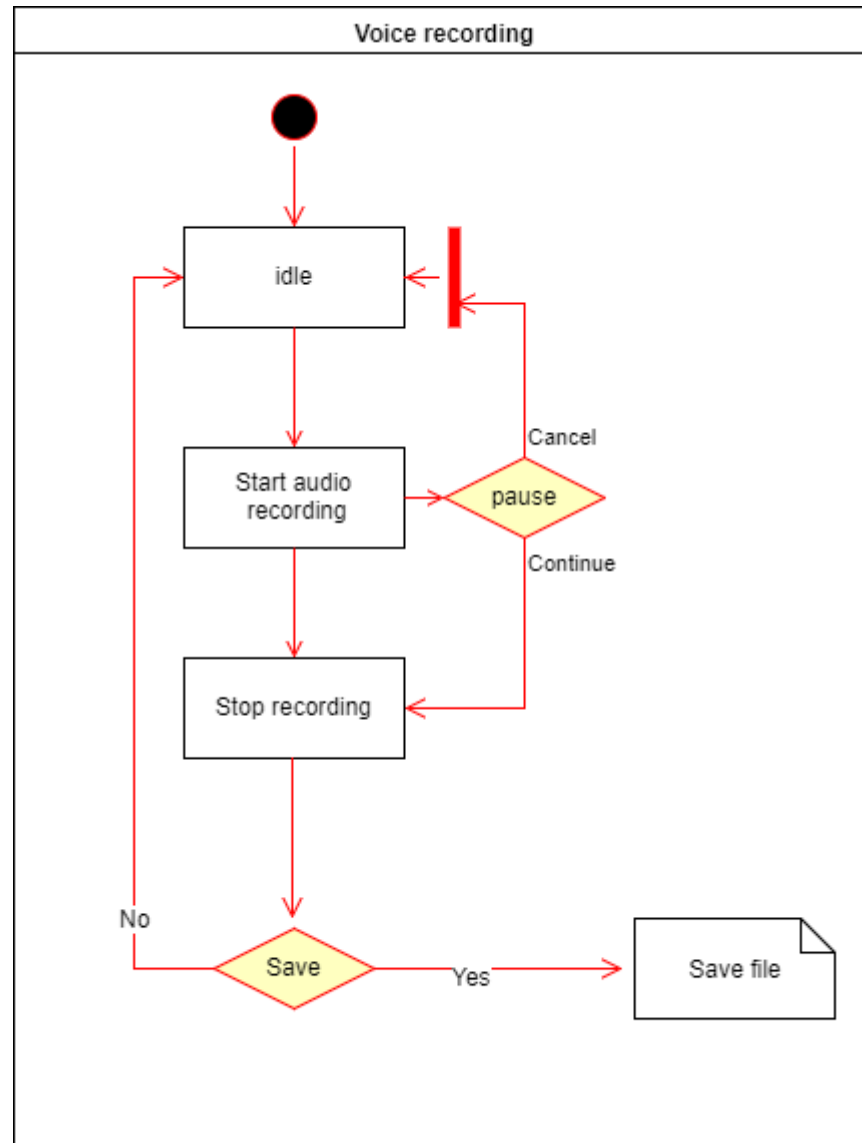


Fig. 2: Audio Recording Activity – This activity shows the workflow of how an audio file is saved into the file system after the user stops the audio recording. While recording, user can pause and restart the recording. The audio file is stored in a specific directory in the Tizen file system.

AUDIO FILE TRANSFER ACTIVITY

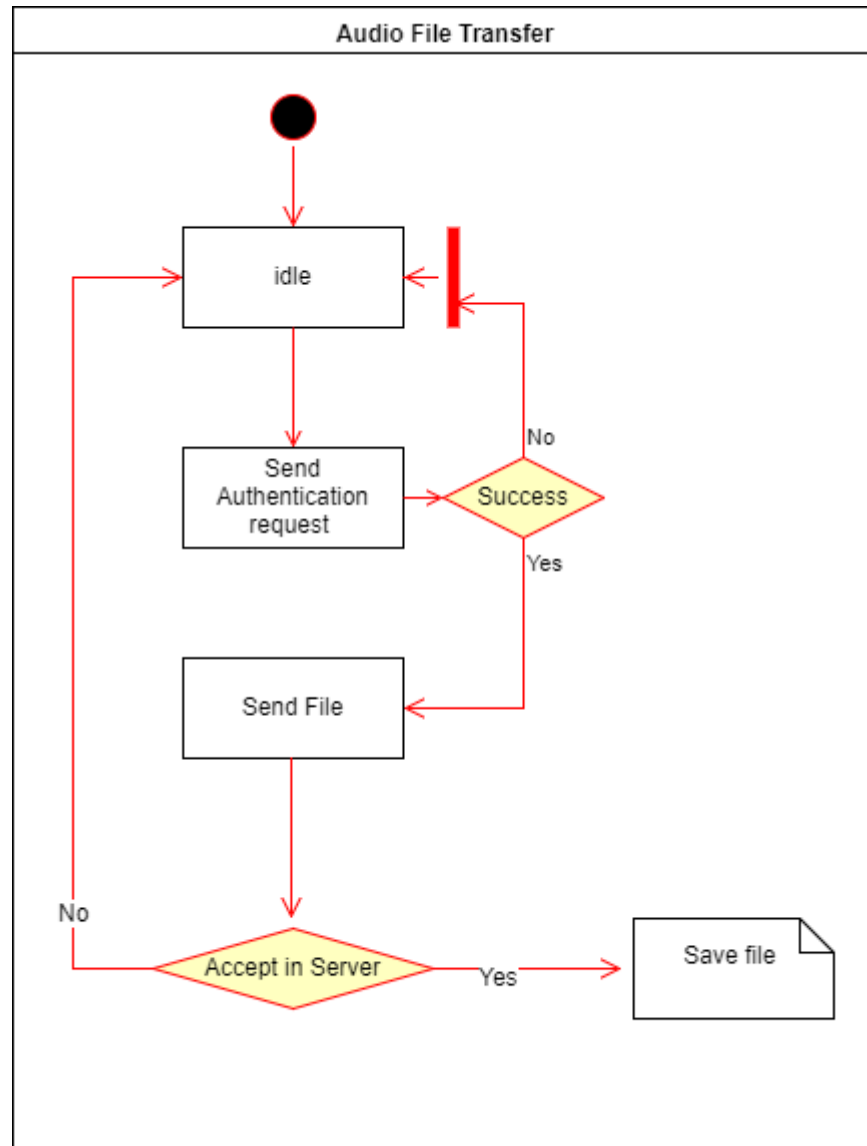


Fig. 3 – Audio File Transfer Activity – This activity shows the workflow of how the audio file is transferred to the server using HTTP POST protocol. This single threaded application first sends an authentication request to the server. Server reads the HTTP header to authenticate the request and stores the **.amr** audio file in the server filesystem.

7. TEST CASES

As part of development and continuous integration, I have tested the entire project with the following testcase scenarios -

- ✓ User record an audio message and press the stop button. The audio file is stored in the specific location.
- ✓ User starts recording the audio file and pauses the recording, and then restarts it after some time. The audio file saved successfully.
- ✓ User starts recording the audio file and pauses the recording, and then cancels it after some time. No audio file is stored.
- ✓ User starts recording the audio file and cancels the recording. No audio file is saved.
- ✓ User starts recording the audio file and finishes the recording. The audio is successfully played from the smartwatch.
- ✓ Multiple audio files are generated for multiple recordings.
- ✓ Smartwatch restart persists the audio file in the Tizen file system.
- ✓ Send the file from smartwatch to server and server saves the file in its local directory.
- ✓ Authenticate the request from smartwatch before processing the audio file.

8. PROJECT STATUS

The table below describes the different modules and the current status of those modules respectively.

| | Module | Status |
|----|--|--------|
| 1. | Develop the UI for the smartwatch application | Done |
| 2. | Record the user's voice on clicking the start button | Done |
| 3. | Add recording pause functionality in the APP | Done |
| 4. | Add stop recording feature in the APP | Done |
| 5. | Store the recorded audio data in a file | Done |
| 6. | Set up the server in local machine | Done |
| 7. | Develop the REST API which accepts the HTTP POST request from smartwatch | Done |
| 8. | Send the file from smartwatch to server | Done |
| 9. | Receive the file from smartwatch and store it in the server | Done |

9. OPEN ISSUES

1. More strict validations can be added both in UI and server side.

10. FUTURE SCOPE

In future, the following features can be added in the project:

1. A better UI with more validations.
2. Send the file on connecting with WIFI.
3. Persist the file in the database rather than storing it in the file system..

REFERENCE

1. <https://stackoverflow.com/questions/tagged/tizen>
2. <https://www.tizen.org>
3. <https://en.wikipedia.org/wiki/Tizen>
4. <http://draw.io>