VISIDA Project Overview

The VISIDA system is a toolkit for the collection and analysis of image-and-voice-base food records. Core functionality requires the installation and use of an Android App and Content Management System (CMS) visible in Figure 1. The CMS is further partitioned into an Angular.js web client, a .NET RESTful API, and a SQL Server.

This document give an overview of the basic steps required for installation of these core features. The software is provided as-is, and assumes a high familiarity with software development.

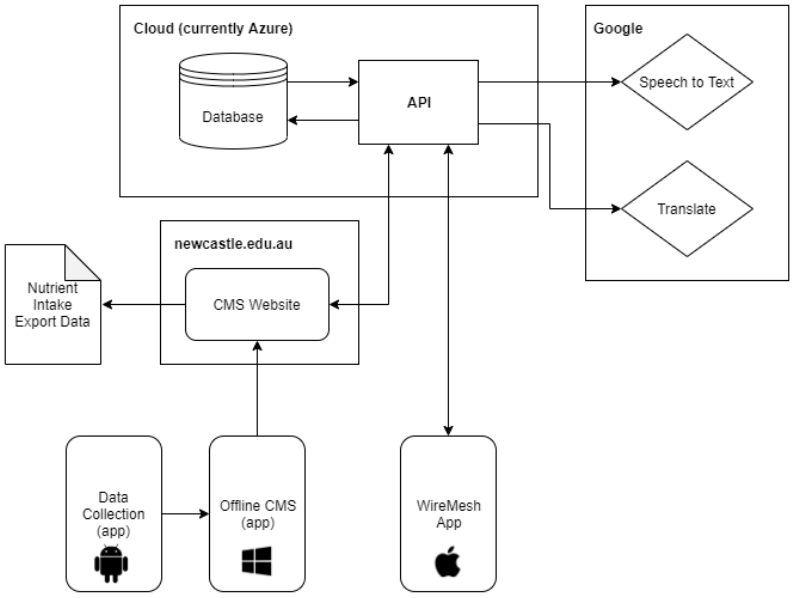

Figure 1. VISIDA system design

VISIDA App

Files related to the Android app are included in *visida_app*. The most recent build is provided as *VISIDA-master-release_31_03_22.apk*. This app was developed in Android Studio, which is recommended for any further updates or changes. Some documentation on updating the localization for a new region is provided in visida_app/Documents.

VISIDA Web Client

Files related to the web client are included in *visida_cms*. The client is developed in Angular.js, and requires Node.js to be installed.

Installation

Install Node.js. Angular.js documentation provides a guide on this. https://docs.angularjs.org/tutorial

Open a terminal in the visida_cms folder and use the Node.js package manager to install the required packages.

```
npm install
```

The client must be served from a HTTP web server. For development purposes, the http-server package works well. Install it with:

```
npm install --global http-server
```

You can then serve the client with:

```
http-server -o
```

Config

The client must be set up to point at the API. Config settings can be found in /package.json. Edit baseUrl inside the configSettings constant to the address of your API, retaining the /api suffix.

```
app.constant('configSettings', {
    'baseUrl': 'http://localhost:39548/api'
});
```

VISIDA API

Files related to the REST API and SQL database are included in *visida_backend*. The API is developed with AspNet .NET.

Installation

Open the .sln solution file in Visual Studio 2019 or higher. If the packages do not download and install automatically, they process can be run by opening the Package Manager console and running:

```
dotnet restore
```

SQL Server

The API uses Entity Framework migrations to create the database schema.
Create an SQL database and update the connection string at *visida_backend/VISIDA_API/Web.config* to point at the SQL server.

- "Initial Catalog" is used to reference the database name.
- The name must remain as VISIDA_APIContext

```
<connectionStrings>
  <add name="VISIDA_APIContext" connectionString="Data Source=DESKTOP-M2AQCB7\SQLEXPRESS; Initial Catalog=visida-api-test;
      Integrated Security=True; MultipleActiveResultSets=True;" providerName="System.Data.SqlClient" />
</connectionStrings>
```

Run the migrations to update the database by opening the Package Manager console and running:

```
Update-Database
```

SQL servers other than T-SQL will require the Entity Framework provider to be updated.

Config

Running the migrations to create the database will insert a default LoginUser with username = **admin** and password = **admin**. It is advised to update this to a secure password, and create new users with lower permission levels through the CMS (when operational) for regular usage of the system.

Use of Google Cloud Services will require setup and configuration of the appropriate APIs through the Google Cloud Platform. Once a .json key file has been created and downloaded for your project, you should update the GOOGLE_APPLICATION_CREDENTIALS environment variable to reference the key file's folder location.