PAAS

Pronounce as a Service

**System Design Document**

**Design Document**

**Version: V1.0**

**Author: Team**

**Date: 15-May-2022**

Table of Contents

[Introduction 3](#_Toc103645624)

[Problem Statement 3](#_Toc103645625)

[Solution 3](#_Toc103645626)

[User Requirements 3](#_Toc103645627)

[UML Use Cases 3](#_Toc103645628)

[UML Class Diagram 4](#_Toc103645629)

[Authentication 5](#_Toc103645630)

[User Profile. 6](#_Toc103645631)

[Speech to text. 6](#_Toc103645632)

[freeTTS 6](#_Toc103645633)

[Microsoft Speech service. 6](#_Toc103645634)

# Introduction

## Problem Statement

Our names are entwined with our personal identities, often chosen with care and laden with meaning. They can represent who we are, where we come from or who we strive to be. Yet people live with a name they do not feel is their own.

Because they feel compelled to adapt their name or adopt a new name entirely to “fit in”, or because their name is routinely mispronounced.

## Solution

**PAAS – Pronounce as a Service**

**Paas** has the ability to render standard pronunciations from trusted third party software/api and also has the ability to record own preferred pronunciations.

**Paas** has the ability to produce pronunciations from an open source library and also licensed software both, with just a toggle of a switch.

**Paas** provides options for the employee to multiple pronunciations

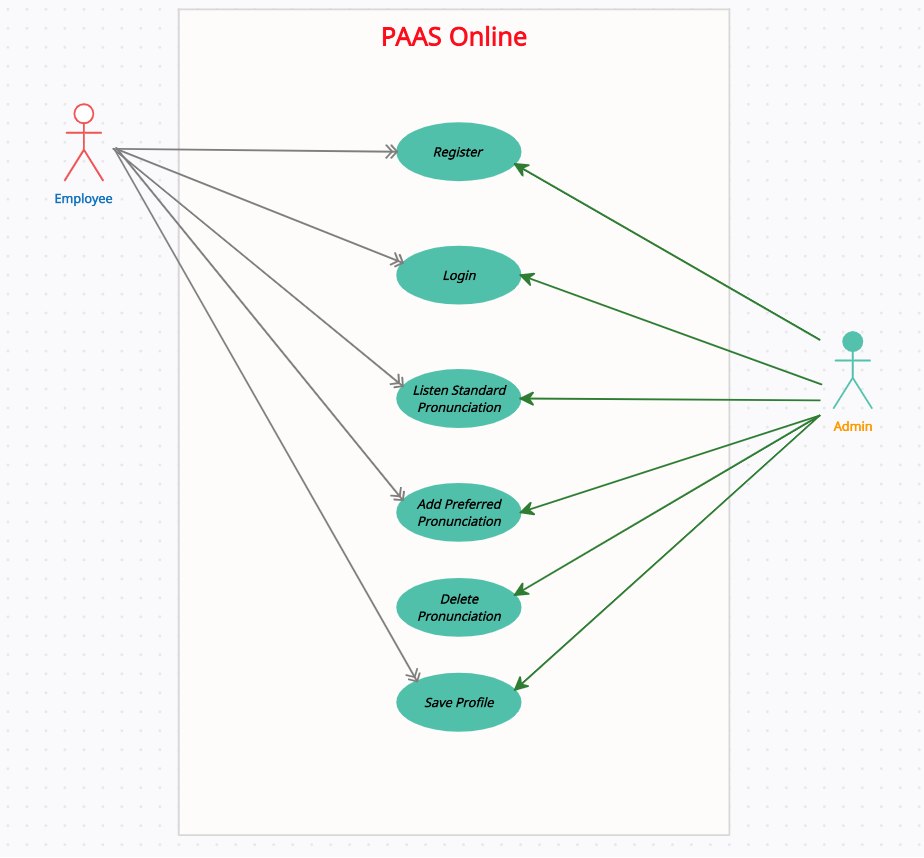
Eg : Rajappa, Primary pronunciation : raajappa , Secondary pronunciation: raaj

## User Requirements

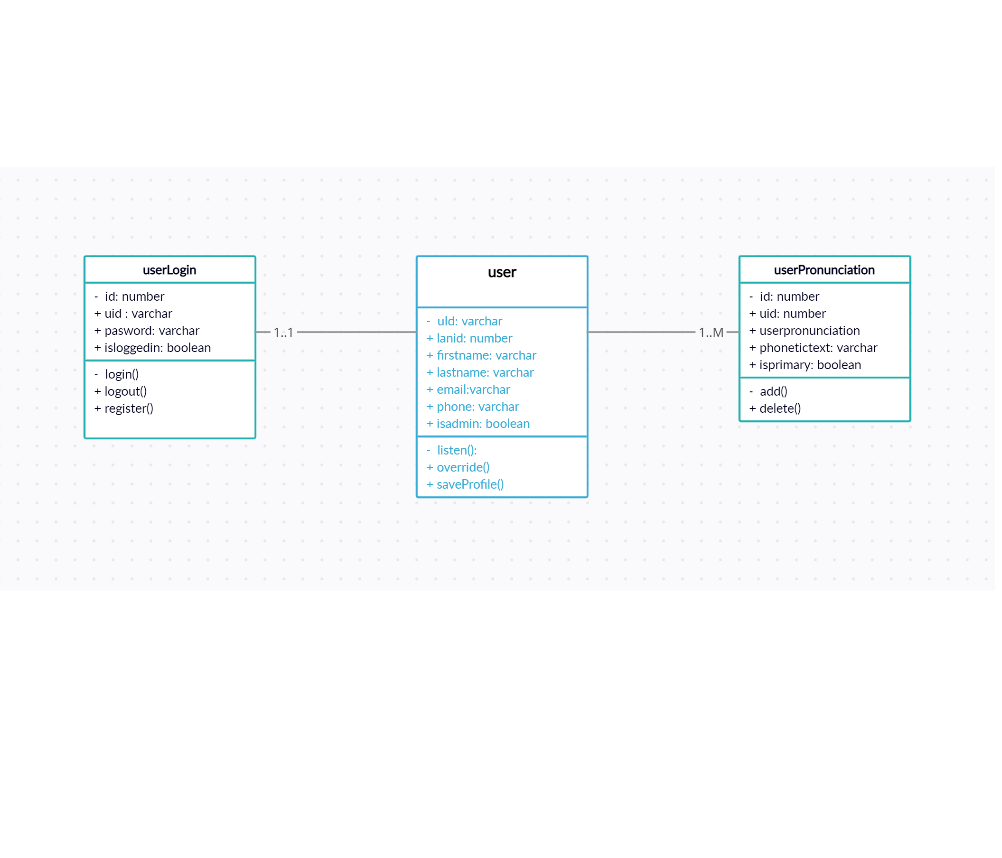
All user requirements are documented in a separate document and uploaded in git under the location marked below.

<https://github.com/Paas-Hack/paas/blob/main/documents/PAAS_UserStories.docx>

## UML Use Cases



## UML Class Diagram



## 

## Authentication

**JSON** **Web Token** is a standard used to create access tokens for an application.

It works this way: the server generates a token that certifies the user identity, and sends it to the client.

The client will send the token back to the server for every subsequent request, so the server knows the request comes from a particular identity. This architecture is effective in modern Web Apps, where after the user is authenticated, we perform API requests either to a REST or a GraphQL API.

A **JWT** is cryptographically signed (but not encrypted, hence using HTTPS is mandatory when storing user data in the JWT), so there is a guarantee we can trust it when we receive it, as no middleman can intercept and modify it, or the data it holds, without invalidating it. API authentication, and server-to-server authorization are best use cases for JWT usage.

## User Profile.

PAAS uses a register functionality to setup users, so we will need to first setup the meta data for the users and then consume the register API, which will encrypt the password and setup the user.

Once user is setup, login will be done through JWT as explained above.

## Speech to text.

In PAAS we envision a scenario where the requirements are specific and achieving them in the best possible way requirements lot of aspects to be factored in so we have a two-way approach in developing this solution.

WE can toggle between a free opensource api or a licensed version of the api to generate speech to text.

Option1: freeTTS

Option2: Microsoft Speech Service.

### freeTTS

Easy Speech2Text will recognize the voice from audio and convert it to text by its machine learning model. Three models for choices.

### Microsoft Speech service.

Microsoft offers speech to text as part of their Azure services. This is a licensed service as part of azure offerings. We need to generate a subscription key and use it on our implementation.

For demo purposes, we have used a standard free subscription, this may cost, if used in the enterprise for large number of requests.

Recorder

User will have the ability to record the preferred pronunciation and store it in the database. The get User api will fetch the recordings, if there is a preferred one.