**AIGS 1008**

**DATA WAREHOUSING & BUSSINESS INTELLIGENCE**

**PROJECT TITLE: SAY MY NAME**

**(Group 1 & 2)**

**Team Members:**

Ninu Basheer

Sanjeev Kandel

Supriya Polangari

Vijay Bhaskar Bonthu

Vysakh Mohan Therattil

Neha Mohammad Parveen

Sonali Jammichedu

Sana Naseem

Olabode Aremu

Vennela

# ABSTRACT

The project aims to enhance respect and inclusivity in academic settings by ensuring accurate pronunciation of culturally diverse names. It features an interactive web interface where students submit their names and IDs and choose phonetic options for correct pronunciation. The backend processes these submissions, offering phonetic suggestions and audio conversions. Unique features include user customization in phonetic selection and a voting mechanism for feedback. This comprehensive approach promotes a more inclusive academic environment by addressing the challenge of name mispronunciation.

Contents

[ABSTRACT 2](#_Toc728177719)

[1. Introduction 3](#_Toc783533918)

[2. Literature Review 4](#_Toc1216283739)

[3. Methodology 5](#_Toc1767684100)

[3.1 Frontend 5](#_Toc957288133)

[3.1.1 Application Flow 5](#_Toc2089015578)

[3.1.2 Framework and Tools used.ools used. 5](#_Toc256474361)

[3.1.3 Libraries and its usage. 5](#_Toc582221077)

[3.1.4 Folder Structure 6](#_Toc998693655)

[3.1.5 Installation and running the project through Angular. 7](#_Toc1817971037)

[3.1.6 User Interface 8](#_Toc81818503)

[3.2 Backend 11](#_Toc1391210364)

[3.2.1 Endpoints 12](#_Toc207074759)

[3.2.1 Audio Conversion with GTTS Module 12](#_Toc707475870)

[3.2.2 Database Storage 12](#_Toc2114791277)

[3.3 Challenges Faced 12](#_Toc1908532784)

[3.4 Error Handling 12](#_Toc1432860715)

[3.5 Security and Permissions 12](#_Toc863031867)

[3.6 Run the Project 12](#_Toc2090695832)

[3.7 Change in file before running 12](#_Toc1113208354)

[4. Results 12](#_Toc1070895208)

[5. Conclusion 12](#_Toc442815929)

[6. Reference Links 12](#_Toc943845209)

# **Introduction**

The project is thoughtfully designed to tackle the challenges associated with accurately pronouncing names originating from diverse cultural backgrounds. Its central objective is not just to recognize every student's name but to ensure that each name is pronounced correctly, creating an atmosphere characterized by respect and inclusivity.

The journey within this project commences with a simple yet crucial step. Students engage with an online form, where they provide essential details about themselves. This initial interaction serves as the foundation for capturing the precise information necessary for achieving accurate pronunciation. This form also consists of an option to select the user’s pronoun.

Following this, students are presented with a range of phonetic pronunciation options for their names. Multiple suggestions for phonetic representations are presented as radio buttons, offering students a variety of choices. This empowers students to select the pronunciation that best aligns with their preferences and cultural background.

The chosen phonetic representation, whether selected from the provided suggestions is prominently displayed as text in the next page. It is presented alongside the student's preferred name and ID, once this information is securely saved in the database. This ensures that the pronunciation preferences along with the student ID of each student are saved in the database uniquely for future reference.

After saving the user’s details in the database, users are encouraged to provide feedback, enabling us to evaluate the user interface's effectiveness in delivering a satisfactory experience.

In addition to this, the application also has one more page namely “Edit/View” page.

One noteworthy aspect of this project is its commitment to proper error handling. Whenever an error code is received from the API response, dedicated error handling pages come into play. These pages are strategically incorporated to gracefully manage and communicate errors to the user, ensuring a smooth and user-friendly experience.

In summary, the project's primary goal is to accurately pronounce names from diverse cultural backgrounds, thereby fostering an environment of respect and inclusivity. This is achieved through a structured process that begins with data collection, offers a variety of phonetic options, and ensures proper error handling to enhance the overall user experience.

# **Literature Review**

Numerous academic institutions, including universities and colleges, have conducted extensive research on projects akin to "Say My Name." These initiatives share a common objective: the accurate pronunciation of names, which aligns with our own work. For instance, "The University of Warwick" has initiated a project aimed at investigating whether a lack of familiarity with the pronunciation and spelling of names might pose barriers to effective teaching and learning [1]. Moreover, we have come across several projects with objectives that closely mirror our own, focusing on providing phonetic renditions of names.

One noteworthy example is the project undertaken by "Nottingham Trent University," which shares significant similarities with our initiative. However, our project distinguishes itself by offering a unique feature: customization. We empower users to tailor the phonetic representation of their names according to their preferences. If the system-generated result does not fully meet their expectations, they have the option to choose their preferred phonetics from a set of provided alternatives. This customization aspect sets our project apart and enhances the user experience, ensuring that individuals have control over the pronunciation of their names.

It is worth noting that our project's concept resonates with many online name generator tools. However, a key differentiator is the comprehensiveness of our database, particularly concerning regional names. Many existing tools struggle to provide accurate pronunciation because they lack a diverse range of words and phonetic data related to regional names. In contrast, our project is actively working to expand its database to encompass a wider spectrum of regional names, enhancing our ability to generate precise and culturally sensitive phonetics. This commitment to inclusivity and accuracy sets our project apart in the realm of name pronunciation solutions.

In summary, our project aligns with various academic studies and similar initiatives, all focused on achieving correct name pronunciation. While some projects share common objectives, our distinctive feature is user customization, allowing individuals to select their preferred phonetics. Additionally, our dedication to enriching the database with regional names distinguishes us from many online name generator tools, which often fall short in providing accurate pronunciations due to limited data coverage.

# **Methodology**

# [**3.1 Frontend**](#_Methodology)

# **3.1.1 Application Flow**

Landing page which offers users to register their email addresses by providing Student ID, Pronoun, Preferred Name, First Name, Last Name. Clicking the submit button will let them register and prompt with recommended phonetic pronunciation of their name.

User will be able to provide feedback, edit their entered phonetics.

# [**3.1.2 Framework and Tools used**](#_Methodology)

Angular, TypeScript, JavaScript, Cascading Style Sheet (CSS), Syntactically Awesome Style Sheet (SCSS – Sassy CSS), Hypertext Markup Language (HTML/ HTML5), Bootstrap, Google Fonts (font service by google)

# **3.1.3 Libraries and its usage**

|  |  |
| --- | --- |
| **Library** | **Usage** |
| Browser Module | Built-in module in angular that exports the functionalities required by the angular app to work. |
| HttpClient Module | Used for making http requests to the backend servers |
| Routing Module | Used to navigate between different components and views in an angular application. |
| Browser Animation Module | Used to add animations to the web applications. |
| Angular Material | Set of visual components that allows us to develop consistent user interface supported by angular team. |
| Toastr Module | Used to add notifications to the angular application. |
| DomSanitizer | Helps to prevent Cross Site Scripting Security bugs (XSS). |
| NgxUi Loader | Customizable loader/ spinner, used when the APIs gives late response |

# **3.1.4 Folder Structure**

**|--src** : Contains source file for the root level application

**| |--app**

**| | |--dialog-module component** : Responsible to generate a popup on the application

**| | |--main component :** Responsible to generate the view for home page of Say My Name application

**| | |--student-edit-view component :** Responsible to generate the view for edit and view page of Say My Name application

**| | |--app component**

**| | |--app routing module.ts** : Contains the route for the overall application

**| | |--app module.ts** : Contains the import for all the modules and librariesused

**| |--assets :** Holds all the images/audio files used in the application

**| | |--images**

**| |--index.html :** This file serves as the entry point for the Say My Name application. Defines the basic structure of your web page, including the necessary scripts, stylesheets, and other resources.

**| |--main.ts :** Bootstraps the root module to run in the browser and triggers the rest of the application source code from that point

**| |--style.scss :** This file applies the styles to the entire application unless you mention styles at a component level

**|--angular.json :** CLI configuration defaults for all the projects in the workspace

**|--package-lock.json :** Provides version information for the installed packages

**|--package.json :** Configures npm package for the dependencies

**|--README.md :** Introductory documentation for the root application

**|--tsconfig.app.json :** Additional configuration file that allows you to adjust your configuration on an app basis i.e. if you have multiple apps running in the same workspace

**|--tsconfig.json :** This file specifies the root files (entry points) and the compiler options required for compiling the project

**|--tsconfig.spec.json** : Responsible for typescript configuration for the application tests

**A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated**

Figure 1 Figure 2

# **3.1.5 Installation and running the project through Angular.**

1. Installation

* Install node.js from [Node.js (nodejs.org)](https://nodejs.org/en)
* Install angular CLI – open the terminal window and run the following command.

**npm install -g @angular/cli**

* Install Node Package Module (NPM), on the command line, run the following command [Downloading and installing Node.js and npm | npm Docs (npmjs.com)](https://docs.npmjs.com/downloading-and-installing-node-js-and-npm)

**npm install -g npm**

* Set the path in the environment variables for NPM.

1. To check whether the installation has been done properly or not, on the command line run the following command.

**ng version**

You should get something like the below image after running the above command.

A screenshot of a computer

Description automatically generated

Figure 3

1. Creating an angular application, use the following command in the terminal.

ng new <project name>

1. To run the frontend, use the below command.

“ng server” or “ng s”

# **3.1.6 User Interface**

A blue screen with white text

Description automatically generated

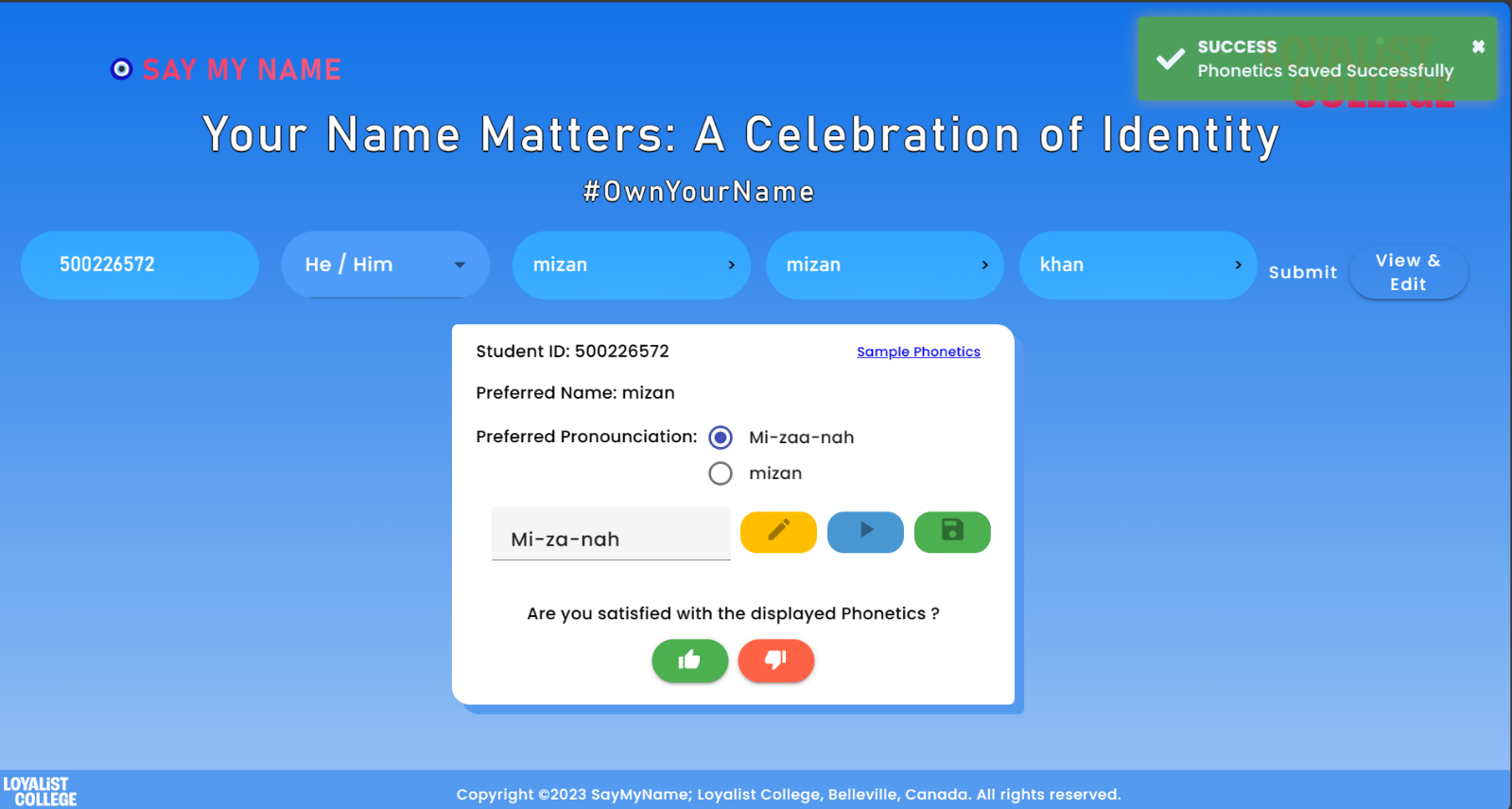
Figure 4: Landing Page

Figure 5: Entering new Phonetics

A screenshot of a computer

Description automatically generated

Figure 6: Viewing and updating the record.

# **3.2 Backend**

In our system, which leverages the Fast API framework for creating a robust and RESTful web service, the process of data collection commences at the frontend through a user-friendly form. This collected data is subsequently transmitted to a dedicated POST API endpoint, where it undergoes meticulous validation to ensure its compliance with the required structure and format standards. The application has been containerized using Docker which will make sure that the App runs on the required environment and will not break on the updates.

Backend Folder of the app contains all the backend Code with its own Dockerfile which is used to setup the environment that is required to run it. While the Frontend named as SayMyName Project Frontend folder contain its own Dockerfile for its fronend requirements and Docker-compose.yml is been used to orchestrate the deployment.

Module named “Pydantic” has been used to perform this data type validations.

The FastAPI application consists of the following components:

1. **Main Script (main.py)**: The main script serves as the entry point for the FastAPI application. It initializes the FastAPI app, defines API endpoints, and starts the UVicorn server.
2. **Model Definitions**: The models.py file contains SQLAlchemy model definitions for the database tables used in the application.
3. **Database Connection (database.py)**: The database.py file establishes the database connection using SQLAlchemy and provides a function to obtain a database session.
4. **Request Models (p\_model\_type.py)**: The p\_model\_type.py file defines Pydantic models for request payloads used in various API endpoints.
5. **Utility Modules**: Additional utility modules such words and handling different languages and deleting scripts if the record is not submitted within 2 mins, respectively.

# **3.2.1 Endpoints**

1. **Ping**
   * **Method:** GET
   * **Path:** /ping
   * **Description:** Endpoint to check the connection to the backend.
2. **Create Student Record**
   * **Method:** POST
   * **Path:** /createpost
   * **Description:** Endpoint to create a new student record in the database.
   * **Request :** {

"first\_name": "string",

"pronoun": "string",

"last\_name": "string",

"preferred\_name": "string",

"student\_id": 0,

"course": "string",

"intake": "Fall",

"year": 0

}

* **Response:** {

"data": {

"student\_id": 0,

"first\_name": "string",

"last\_name": "string",

"full\_name": "string string",

"preferred\_name": "string",

"audio\_binary": "\\xfff344c400118001dc014100010f0f0f3ff800efffffffffffffffffffffffdfffec03ff0030f0",

"pronoun": "string",

"phonetics": [],

"data\_in\_votes\_table": false

},

"results": [],

"status": "success",

"message": ""

}

If the student\_id exists then throws the below response

{

"status": "failed",

"message": "Student ID already exists"

}

1. **Create Selection Record**
   * **Method:** POST
   * **Path:** /selection
   * **Description:** Endpoint to create a selection record for a student's phonetic preference.

* **Request:** {"student\_id": 0,

"name": [

"string"

],

"phonetics\_selection": [

"string"

],

"audio\_selection": "string",

"show": true,

"data\_in\_votes\_table": false

}

* **Response:** {"status":'failed',

"message":"Data is not present in votes table please check data\_in\_votes\_table files in set to false"}

* + **Failed Response:**

{

"status": "success",

"message": ""

}

1. **Get Student Record**
   * **Method:** GET
   * **Path:** /getRecord/
   * **Description:** Endpoint to retrieve student records based on student ID.
   * **Request:** getRecords/?studentID=1
   * **Respons**e: {

"student\_id": 92900217,

"first\_name": "winston",

"last\_name": "mrs",

"preferred\_name": "winston",

"phonetics\_selection": "win-ston",

"pronoun": "He/HIm",

"course": "AIGS",

"intake": "Fall",

"year": 2023,

"show": true

},

1. **Get Student Records**
   * **Method:** GET
   * **Path:** /getRecords/
   * **Description:** Endpoint to retrieve student records based on various filters. (Can be used as Global search for Admin)
2. **Update Student Record**
   * **Method:** PUT
   * **Path:** /update
   * **Description:** Endpoint to update an existing student record.

* **Request:**

{ "student\_id": 0, "first\_name": "string", "pronoun": "string", "last\_name": "string", "preferred\_name": "string", "course": "string", "intake": "Fall", "year": 0, "phonetics\_selection": "string"}

* + **Response:**

{

"status": "success",

"message": "updated record successfully"

}

If there is any error during the update below will be the response  
   
return {"message": e,

"status": "failed"}

1. **Submit User Feedback**
   * **Method:** POST
   * **Path:** /userfeedback
   * **Description:** Endpoint to submit user feedback.

* **Request:**  
  { "student\_id": 0,

"userfeedback": "string"}

* **Response:**

{

"status": "success",

"message": ""

}

{

"status": "failed",

"message": "incorrect details received or feedback for this user is already exist."

}

1. **Delete Student Record**
   * **Method:** DELETE
   * **Path:** /deleterecord
   * **Description:** Endpoint to delete a student record from the database.

* **Request: {"**student\_id": 0

}

* **Response body:**

{

"status": "success",

"message": "Deleted record successfully"

}

If there is no record then you will get below response

{

"status": "success",

"message": "Record with 0 doesn't exist in the system "

}

1. **Get Phonetic Audio**
   * **Method:** GET
   * **Path:** /getaudiophonetics
   * **Description:** Endpoint to retrieve audio files for phonetic names.
   * **Request:** { "phonetics\_name": "wins-ton"}
   * **Response:** Stream audio file
2. **Get Audio**
   * **Method:** GET
   * **Path:** /getaudio
   * **Description:** Endpoint to retrieve audio files for preferred names.
   * **Request:** { "phonetics\_name": "wins-ton"}
   * **Response:** Stream audio file

# **Audio Conversion with GTTS Module**

Get Audio and Get Phonetics Audio uses GTTS (Google Text-to-Speech) module. Here, the text data is transformed into an audio format (wav format), allowing for the generation of spoken pronunciations. Which can also be used to validate if the manual entry of the phonetic sounds.

# **Database Storage**

After successfully traversing all these processing stages, the system compiles the information into json format. This compiled data is then securely stored within a Postgres database residing on the backend server.

A screenshot of a computer

Description automatically generated

￼

Figure 7: Database Schema

Our database architecture is designed with Five core tables:

* **Student\_Data**: This table is dedicated to maintaining comprehensive student profiles. It stores essential student information, including their names, Student IDs, and other relevant details.
* **Pronunciation:** The second table is dedicated to logging the names selected by students. This logging is of paramount importance for our recommendation, which evaluates name popularity based on the frequency of selections by users. This data assists in making informed suggestions to users in the future.
* **Userfeedback:** This Table is dedicated to log user feedback of the app.
* **Phonetics\_table:** This Table contains all the unique names and Phonetic pronunciation which are captured during the Data warehousing Project.
* **Votes:** Capture all the names of the students and unique phonetic Pronunciation of the names, this table also capture the frequency of the phonetic pronunciation so that this can be used to recommend different Pronunciations of the name.

The database effectively serves as a repository for all the data that is required for this Project. This ensures the availability and accessibility of this data for future retrieval and analysis, contributing to the continued improvement of the system and enhancing its ability to provide accurate and culturally sensitive pronunciations of student names.

# **Challenges Faced**

During project development phase, we encountered several noteworthy challenges related to the generation of phonetic sounds, particularly when dealing with specialized symbols and regional names. While our system demonstrated a high degree of accuracy in generating phonetic representations for English names, it became evident that handling regional names introduced a layer of complexity due to the rich diversity of linguistic patterns.

One of the primary challenges we faced was selecting a suitable phonetic library and devising a method to effectively segment words, especially regional names, into their constituent phonetic components. The diverse array of languages and dialects associated with regional names meant that there was no one-size-fits-all approach. Consequently, developing a system that could adapt to these variations proved to be a formidable task.

Additionally, we encountered discrepancies between the audio output and the corresponding phonetic text representation. While the audio pronunciation often proved to be accurate, the text representation of the phonetics did not consistently align with the audio output. This highlighted the need for enhanced precision and consistency in the phonetic representation process.

These challenges underscore the intricate nature of our project, as it seeks to navigate the complexities of linguistic diversity while striving for accuracy and precision in providing phonetic pronunciations. As we continue to refine our system, we are actively worked to address these challenges, ensuring that it can effectively and sensitively handle regional names and deliver accurate phonetic representations consistently.

# **Error Handling**

The Fast API application implements error handling to ensure robustness and reliability. It utilizes HTTP status codes and exception handling mechanisms to handle errors gracefully and provide informative error messages to clients.

# **Security and Permissions**

The application implements Cross-Origin Resource Sharing (CORS) to allow connections from specified origins. Access control mechanisms can be configured to restrict access to certain endpoints based on user roles and permissions.

# **Run the Project**

1. Download Docker desktop.
2. Once installed, login to docker.
3. git clone  <https://github.com/VijayBonthu/SYM_final_project.git>
4. Navigated to the path where it is cloned in command Prompt.
5. in the command prompt run **docker-compose up --build** from the root of the project.
6. The above command should download all the required dependencies and start frontend, backend and postgres DB.
7. If you have installed postgres before it might cause an issue with the ports, if you are facing issue with failed postgres server issue follow **step 10** to fix it. Once the below steps are done run **docker-compose up --build**
8. to stop the container **docker-compose down**.
9. to start the containers again **docker-compose up** if there are no changes to your program files. if you have changes then you have to run **docker-compose up --build**.
10. If you having issues accessing the database from other GUI's it is possible you have already an instance of postrges in your machine and it's not letting you access it. Press **Win + R** to open the Run dialog. Type **services.msc** and press Enter. This will open the Services window. Scroll down to find the PostgreSQL service. It may be named something like **"postgresql-x64-"**. Right-click on the PostgreSQL service and select **"Stop"**.
11. If You see the below screen and highlighted since shows up with out any error, you will be able to access your application.

A screenshot of a computer program

Description automatically generated

# **Change in file before running**

1. To allow your front end to connect, you need to add the port of your frontend application to backend **main.py** in **CORS section**.
2. change your DB connection url ports and password in **docker-compose.yml** in **db** section and **Service** section.
3. make sure the same **DB url** is used in **backend database.py**

# **Results**

The overarching goal of this project is to contribute to the creation of more inclusive and welcoming academic environments by addressing the familiar challenges associated with the pronunciation of culturally diverse names. In academic settings, where individuals from various backgrounds come together, correct name pronunciation is vital for fostering respect and inclusivity.

To achieve this goal, our project provides an intuitive web interface that allows students to easily submit their names and corresponding IDs. What sets our approach apart is the option for students to customize the phonetic pronunciations of their names, tailoring them to their preferences.

Our project architecture is structured to ensure security, error resilience, and a user-friendly experience. Angular frontend, responsible for user interactions, handles data securely and incorporates robust error handling mechanisms. This ensures that users have a smooth and hassle-free experience while interacting with the system.

On the backend, the project relies on Fast API, a powerful framework, to handle the data processing pipeline. This includes modules for phonetic transformation, audio conversion, and database storage. The Postgres database serves as the repository for all this valuable data, securely storing user information and their phonetic preferences.

One unique aspect of our system is the incorporation of a 'Voting' mechanism. This feature allows users to provide feedback and vote on the preferred name pronunciations. Over time, this feedback loop contributes to the improvement of recommended name pronunciations, making them more accurate and culturally sensitive.

In summary, our comprehensive approach ensures that names are pronounced accurately and respectfully, thereby promoting diversity and inclusivity in educational spaces. By combining an intuitive user interface, robust data handling, and an adaptive feedback mechanism, our project contributes to a more harmonious and respectful educational environment, where everyone's name is recognized and pronounced correctly.

# **Conclusion**

Our project successfully addresses the challenge of accurately pronouncing names from diverse cultural backgrounds in academic settings, emphasizing respect and inclusivity. It commences with students providing their names and IDs through an online form, a critical step for precise pronunciation. The system then presents phonetic options, allowing students to select the most accurate representation of their name.

Technically, the project leverages Angular for front-end development, Python for programming, Fast API and Uvicorn for the server, SQL Alchemy and psycopg2 for database interactions, and GTTS. The database is managed using Postgres, and Git is utilized for version control. Docker is used for containerizing the environment with required specifications to run the application which includes all the libraries and modules

In conclusion, our project stands as a comprehensive solution that seamlessly integrates front-end and back-end technologies to enhance the pronunciation of names in educational institutions. This endeavour not only bridges communication gaps but also fosters a more inclusive and respectful academic environment. Future developments could include a feature for users to enter phonetics as text, providing a more personalized experience. Additionally, integrating play buttons alongside phonetic suggestions would offer an immediate auditory confirmation, further enriching user interaction and effectiveness. These improvements would significantly augment the tool's usability and precision.

# **Reference Links**

1. <https://warwick.ac.uk/services/dean-of-students-office/community-values-education/saymyname/research/>
2. <https://www.ntu.ac.uk/research/groups-and-centres/projects/say-my-name-experiences-and-impacts-of-pronunciation-of-names-in-higher-education-in-contexts-of-culturally-diverse-student-identities>
3. <https://us.pg.com/the-name-generator/>
4. <https://www.nameshouts.com/>