Ana Marinic
Hannah Whitely
Micheala Mulgrew
Juli Pakai

Code First Girls Degree Summer 2022 Specialisation Project Report

Linguistics Group

Table Of Contents | Page Number

Ana Marinic
Hannah Whitely
Micheala Mulgrew
Juli Pakai

1. Introduction

1.1. Aims and Objectives

The aim of the project was to create a tool that would automate the use of the Phonetic Features Chart[1] by Jason Riggle. This way, the users of our programme would save time by just having to input their phonemes instead of manually searching the chart.

1.2. Roadmap of the Report

First, the background of the project will be outlined in the report. Second, the specifications of the project and the planning and design stages will be explained. Third, the implementation stage will be outlined, including the development and the roles of the project team members. Fourth, the testing strategy of the project will be explained. Finally, a conclusion about the project will be drawn.

2. Background

2.1. Linguistic Background

The phonetics Feature Chart is mainly used by phonologists (and by extension phoneticians) to find the smallest natural group of features that can be used to characterise a set of phonemes[2]. This is relevant to the discipline of phonology to study language change, morphophonology, and also for the implementation of phonological rules in synthesised speech.

2.2. Product Description

We have built a tool that automates the use of the Phonetic Features Chart. We have also incorporated the availability to choose a language, which limits the pool of phonemes that need to be compared to only those present in the specified language. This is especially important when working on specific languages, and since there are no separate feature charts for separate languages, this makes the determination of common features easier than ever. No time needs to be wasted on searching for common features manually anymore while also minimising the risk of mistakes.

The user only needs to choose a language and select the phonemes they want to find the common features of - the program calculates the smallest common group those features can be characterised with in the specified language. If the phonemes do not form a natural group, the next

[1] Phonetics Features Chart by Jason Riggle (current version 12.12)
http://www.artoflanguageinvention.com/papers/features.pdf

[2] For more information on phonetic features refer to Chapter 12 (What is a possible language: Distinctive features, pp. 254–274) of Zsiga, E. C. (2013) The sounds of language: An introduction to phonetics and phonology. Oxford: Blackwell.

closest smallest group is returned. If a selected phoneme does not exist in the specified language, the program lets the user know so they can correct their mistake.

An example would be putting in "s" and "z" with English as the selected language. The program would return one of three possible combinations, depending on which one it finds first:

1. Dorsal: True, Strident: True
2. Strident: True, Continuant: True
3. Strident: True, Delayed Release: True

2.3. Project Overview

The project was carried out by a team of four members: Ana Marinic, Hannah Whitely, Micheala Mulgrew, and Juli Pakai. After considering several other project ideas, a project that was achievable while still challenging was chosen. It was important to the team to choose a project that solves a real-life problem. The requirements were discussed via zoom call, and the project team made sure that everyone understood what the program needed to be able to do. The team used Slack and Zoom to communicate. GitHub was used as a remote repository to share the code.

3. Specifications and Design

3.1 Requirements

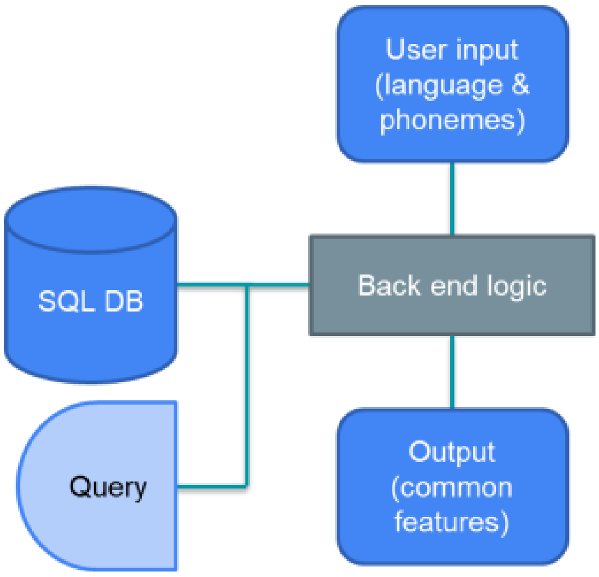
The requirements for the project were as follows:

- to create a database that would store the Phonological Features Chart
- to connect this database to Python
- to create functions that would aid in finding the smallest natural group and the smallest common feature description of the specified group
- to create a visually pleasing HTML user-interface
- to connect the HTML interface to Python using Flask
- as it was a project requirement, we also needed to use an API.

3.2. Design and Architecture

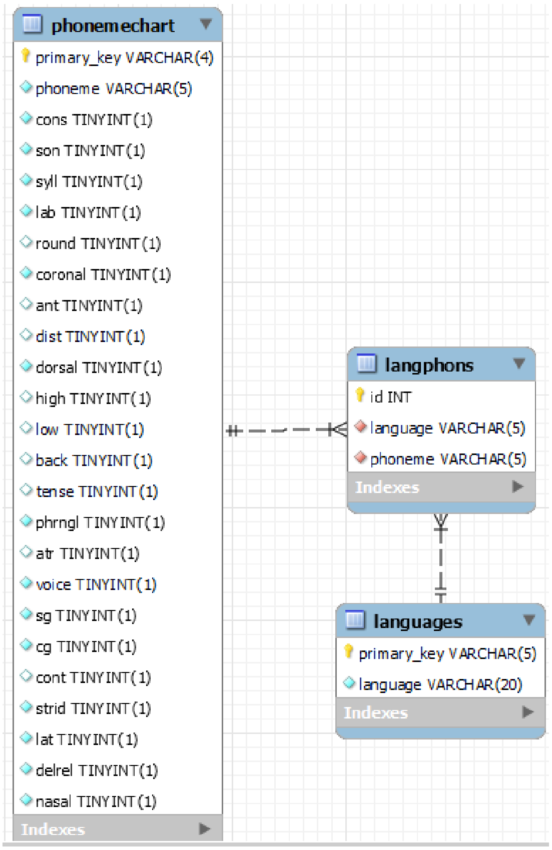
The architecture design of the project is outlined in Figure 1.

Figure 1: Architecture design of the program



Furthermore, the architecture of the MySQL database can be seen in Figure 2.

Figure 2: MySQL database architecture



4. Implementation and Execution

4.1. Project Roles

Micheala dealt with documentation for the project, including the project report and presentation. Juli constructed the database in MySQL, wrote functions in Python, and tested in the backend. Hannah worked on creating a connection to the SQL database using mysql.connector in Python. She also worked on the use of APIs. Ana worked on the use of APIs and Flask implementation using Python. Juli and Ana worked together on frontend development using HTML and CSS.

4.3. Tools and Libraries

We implemented our product using Python for the backend, MySQL for the database, Flask to connect the backend and frontend, and HTML combined with CSS for frontend development.

We used the mysql.connector module in Python to connect the database to Python.

For the Python functions we used the itertools module to create combinations of features. A snippet of the code using itertools can be seen below:

```
while smallest_common_group != phonemes_in_group:
    for i in range(len(common_feats)):
        if i >= 2:
            for combination in list(map(dict,
itertools.combinations(common_feats.items(), i))):
                smallest_common_group.clear()
                for k in range(len(feats_table)):
                    count = k
                    if combination == (dict(combination.items() &
feats_table[k].items())):
                        phoneme = "{}".format(feats_table[k]["primary_key"])
                        smallest_common_group.append(phoneme)
                    if count == (len(feats_table)-1) and smallest_common_group
== phonemes_in_group:
                        smallest_feature_group.update(combination)
                        print(smallest_feature_group)
                        return smallest_feature_group
```

To use an API we used The Cat API to fill the empty space on the output page of the HTML.

4.4. Implementation Process

The implementation went smoothly in general, but the team ran into some difficulties about how to deal with the input in the HTML. The Python function worked by using primary keys, but there was no way for the user to know what these primary keys were. The solution was a checkbox form created in HTML that passed the primary keys as values while still displaying IPA which the user would be familiar with. A snippet of the HTML is below:

```
    <label class="check">
        <input type="checkbox" name="phon_check" value="u5">
        <span>y</span>
    </label>
    <label class="check">
        <input type="checkbox" name="phon_check" value="i4">
        <span>I</span>
    </label>
```

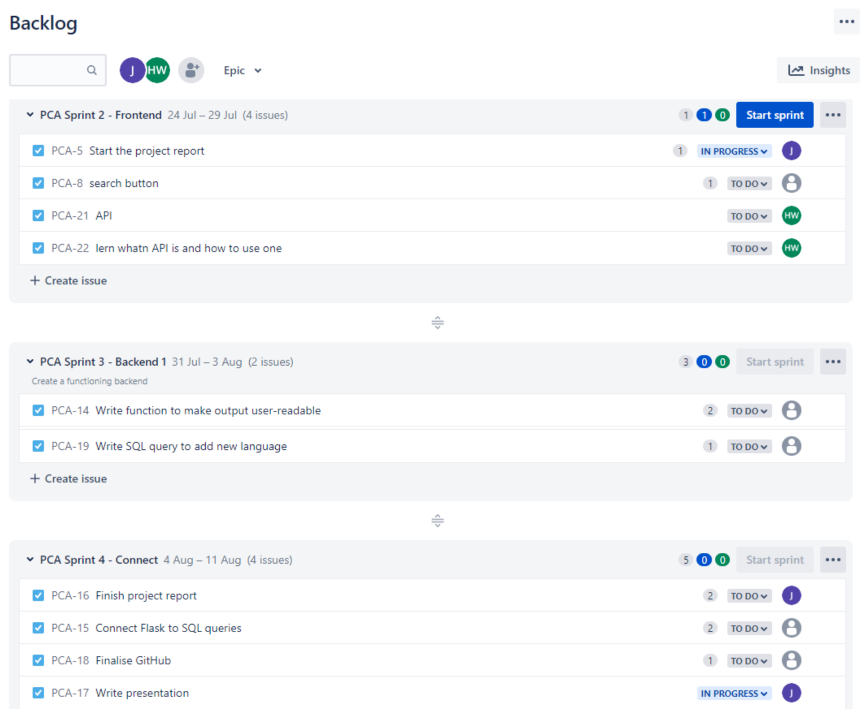
We also struggled with finding a way to use an API as the initial way of wanting to use an API for the Phonological Features Chart or as a way to access the languages did not work out due to lack of resources or hard-to-deal-with output. Therefore, the team decided that The Cat API should be used to fill the empty space on the output HTML. This displays a random picture of a cat when the user uses the tool.

Furthermore, for some reason, the team members divided the roles not according to what they know best. Juli, who knows web development dealt with backend, while Ana, who has previous experience working in Python, dealt with the frontend. In the end, this didn't cause any problems as the team members were able to help each other, this proved to be an excellent learning process.

4.5. Agile Development

The team used Jira project tracking software to distribute tasks among the team and to keep track of what has to be done. The project was divided into sprints. Milestones and deadlines were set using Jira. This can be seen in Figure 3.

Figure 3: Jira backlog of the project in the middle of the development phase



Backlog

PCA Sprint 2 - Frontend 24 Jul – 29 Jul (4 issues) 1 1 0 Start sprint

- PCA-5 Start the project report 1 IN PROGRESS J
- PCA-8 search button 1 TO DO
- PCA-21 API TO DO HW
- PCA-22 lern whatn API is and how to use one TO DO HW

PCA Sprint 3 - Backend 1 31 Jul – 3 Aug (2 issues) 3 0 0 Start sprint
Create a functioning backend

- PCA-14 Write function to make output user-readable 2 TO DO
- PCA-19 Write SQL query to add new language 1 TO DO

PCA Sprint 4 - Connect 4 Aug – 11 Aug (4 issues) 5 0 0 Start sprint

- PCA-16 Finish project report 2 TO DO J
- PCA-15 Connect Flask to SQL queries 2 TO DO
- PCA-18 Finalise GitHub 1 TO DO
- PCA-17 Write presentation IN PROGRESS J

5. Testing and Evaluation

5.1. Testing Strategy

During the development of the functions in Python, a Test-Driven Development strategy was employed to make sure that all the outputs and errors are correct. All three functions were thoroughly tested before deployment.

An example of a unit test can be seen below:

```
class TestComparePhonemes(TestCase):

    def test_compare_phonemes_features(self):
        all_expected = [({'dorsal': False, 'strid': True}, ['s', 'z', 'f',
'v']),
                        ({'strid': True, 'cont': True}, ['s', 'z', 'f', 'v']),
                        ({'cont': True, 'strid': True}, ['s', 'z', 'f', 'v']),
                        ({'delrel': False, 'strid': True}, ['s', 'z', 'f',
'v']),
                        ({'strid': True, 'dorsal': False}, ['s', 'z', 'f',
'v']),
                        ({'strid': True, 'delrel': False}, ['s', 'z', 'f',
'v'])]
        language = "en"
        result = backend.compare_phonemes(language, "s", "z", "f", "v")
        in_list = False
        print(result)
        if result in all_expected:
            in_list = True
        self.assertTrue(in_list)
```

5.2. Functionality and User Testing

In addition to unit testing, the program was comprehensively tested by the team members to ensure that it works as expected.

5.3. System Limitations

The program is limited by the fact that it is only usable if the user has all the required dependencies installed and knows how to run a Flask application. This problem could be solved by hosting the application online. Due to the complexity of implementation, another issue currently is the limited offer of languages. Namely, Spanish, English, and Hungarian are the only languages currently offered.

6. Conclusion

Overall, the development of the product went mostly smoothly, with only minor hiccups. The next steps are to maintain the product, add additional features, and potentially deploy it online so a wider population can use it.