Coding Guidelines

Guidelines:

General Guidelines

- Variables should be formatted using camelCase(Example: localVariable)
- Ensure that different features are being pushed to different branches to avoid potential conflicts
- Performs tests on code pushed to branches prior to merging with the main branch
- Variable and function names should be meaningful and descriptive to help inform an unknown user of their purpose (Example: var1 vs teacherName)
- Ensure consistent indentation throughout the code to help improve the readability of the code
- Comment code if it is difficult to understand
- Avoid lengthy functions

Javascript, CSS, and SCSS Guidelines

- The Prettier coding guideline is being enforced on all javascript, CSS, and SCSS files via the use of a pre-commit hook. The pre-commit hook makes changes to the files when a team member attempts to commit the files to the repository

Python Guidelines

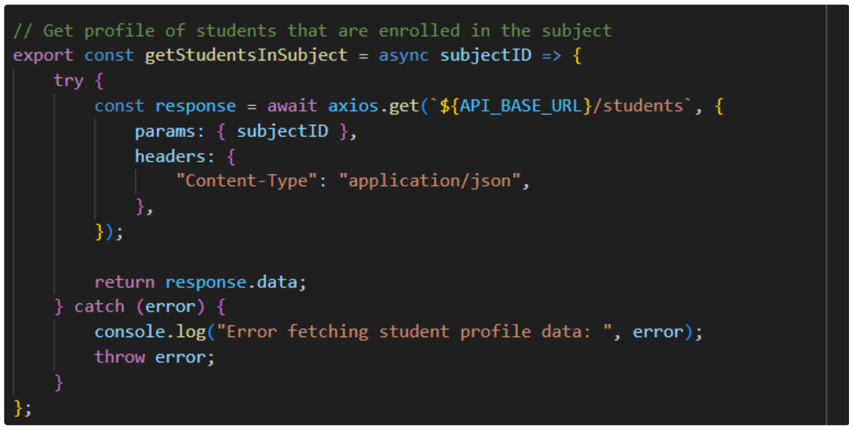
- PEP8 is being enforced on all Python files in the back end via the use of autopep8 and a pre-commit hook. Using the pre-commit hook when files are committed changes are made to the files

Database Guidelines

- Ensure SQL Functions are capitalized to help them distinguish them from table or field names (Example: SELECT)
- Ensure the first letter of field names is capitalized (Example: FirstName)

Example

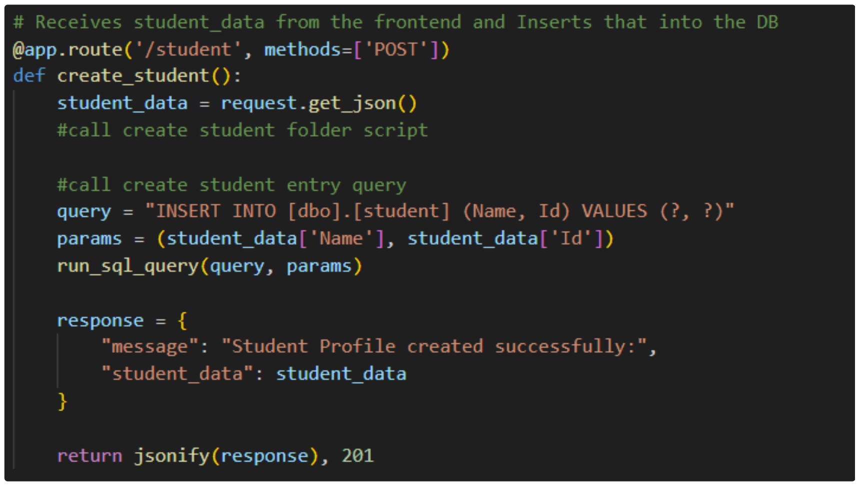
Formatted JavaScript Code



```
// Get profile of students that are enrolled in the subject
export const getStudentsInSubject = async subjectID => {
    try {
        const response = await axios.get(`${API_BASE_URL}/students`, {
            params: { subjectID },
            headers: {
                "Content-Type": "application/json",
            },
        });

        return response.data;
    } catch (error) {
        console.log("Error fetching student profile data: ", error);
        throw error;
    }
};
```

Formatted Python Code



```
# Receives student_data from the frontend and Inserts that into the DB
@app.route('/student', methods=['POST'])
def create_student():
    student_data = request.get_json()
    #call create student folder script

    #call create student entry query
    query = "INSERT INTO [dbo].[student] (Name, Id) VALUES (?, ?)"
    params = (student_data['Name'], student_data['Id'])
    run_sql_query(query, params)

    response = {
        "message": "Student Profile created successfully:",
        "student_data": student_data
    }

    return jsonify(response), 201
```