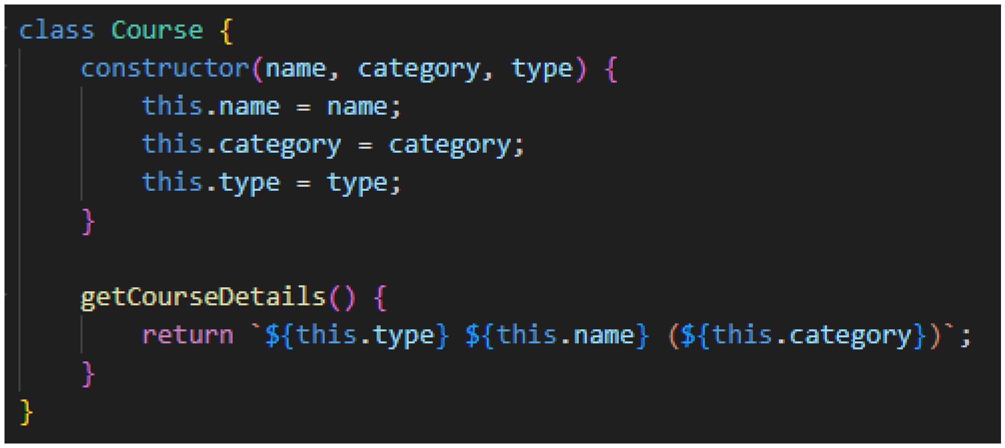
Criterion C: Development

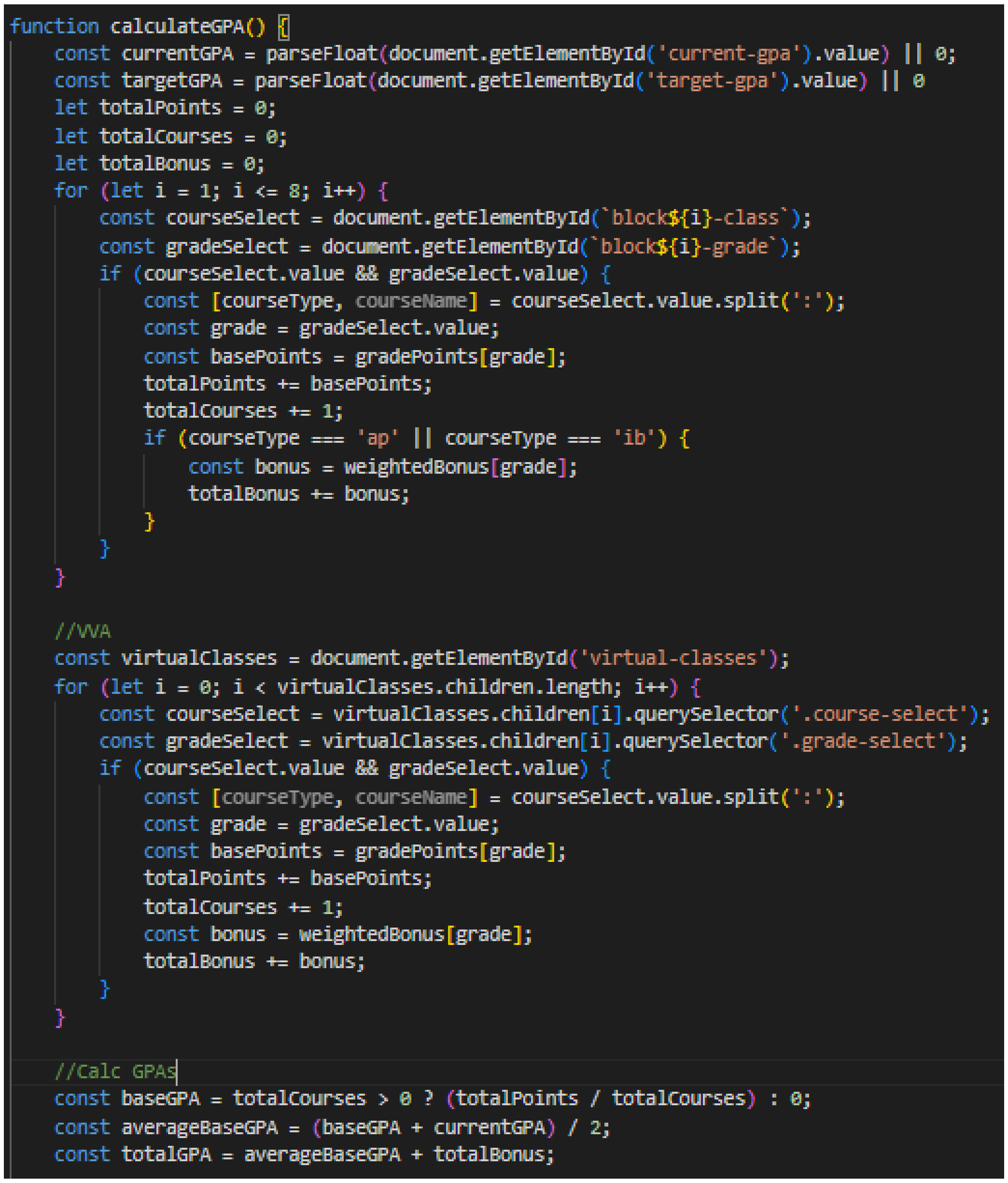
1. Course Class



```
class Course {
    constructor(name, category, type) {
        this.name = name;
        this.category = category;
        this.type = type;
    }

    getCourseDetails() {
        return `${this.type} ${this.name} (${this.category})`;
    }
}
```

Explanation: The Course class encapsulates course details, promoting object-oriented programming by organizing course data into manageable objects. This code shows the constructor class that creates and initializes an object instance of a course by assigning it a name, category, and type which is whether or not the class is weighted. There is also a getCourseDetails() function which can be called to return these values and details of any course object instance.

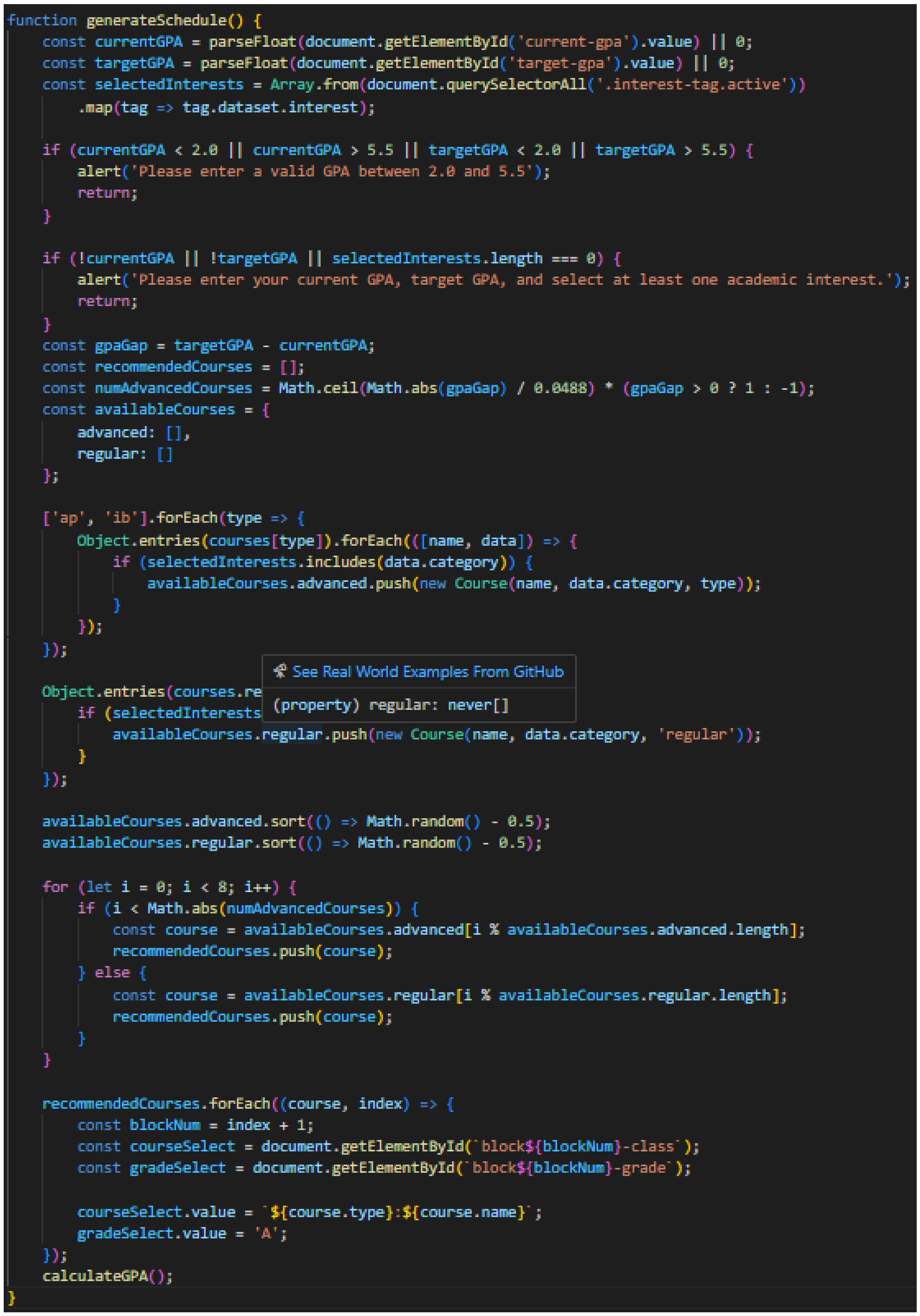
2. CalculateGPA Function



```
function calculateGPA() {
    const currentGPA = parseFloat(document.getElementById('current-gpa').value) || 0;
    const targetGPA = parseFloat(document.getElementById('target-gpa').value) || 0
    let totalPoints = 0;
    let totalCourses = 0;
    let totalBonus = 0;
    for (let i = 1; i <= 8; i++) {
        const courseSelect = document.getElementById(`block${i}-class`);
        const gradeSelect = document.getElementById(`block${i}-grade`);
        if (courseSelect.value && gradeSelect.value) {
            const [courseType, courseName] = courseSelect.value.split(':');
            const grade = gradeSelect.value;
            const basePoints = gradePoints[grade];
            totalPoints += basePoints;
            totalCourses += 1;
            if (courseType === 'ap' || courseType === 'ib') {
                const bonus = weightedBonus[grade];
                totalBonus += bonus;
            }
        }
    }

    //VVA
    const virtualClasses = document.getElementById('virtual-classes');
    for (let i = 0; i < virtualClasses.children.length; i++) {
        const courseSelect = virtualClasses.children[i].querySelector('.course-select');
        const gradeSelect = virtualClasses.children[i].querySelector('.grade-select');
        if (courseSelect.value && gradeSelect.value) {
            const [courseType, courseName] = courseSelect.value.split(':');
            const grade = gradeSelect.value;
            const basePoints = gradePoints[grade];
            totalPoints += basePoints;
            totalCourses += 1;
            const bonus = weightedBonus[grade];
            totalBonus += bonus;
        }
    }

    //Calc GPAs
    const baseGPA = totalCourses > 0 ? (totalPoints / totalCourses) : 0;
    const averageBaseGPA = (baseGPA + currentGPA) / 2;
    const totalGPA = averageBaseGPA + totalBonus;
```

Explanation: This function computes the GPA by considering course grades and weighted bonuses, updating the display with calculated values. It is efficient because it considers Virtual Virginia classes, calculates base GPA by averaging all course grades, and adds a weighted bonus when needed. It handles dynamic addition/removal of Virtual Virginia classes. The code is suitable because it accurately implements the school's specific GPA calculation rules while being flexible enough to handle varying classes. This is all stored in the function CalculateGPA() and variables that store data from the form, loops that are able to go through every single class, and if else statements that decide whether or not a class is weighted, and if courseSelect.value is equal to gradeSelect.value.

3. Generate Schedule Function


See Real World Examples From GitHub
(property) regular: never[]

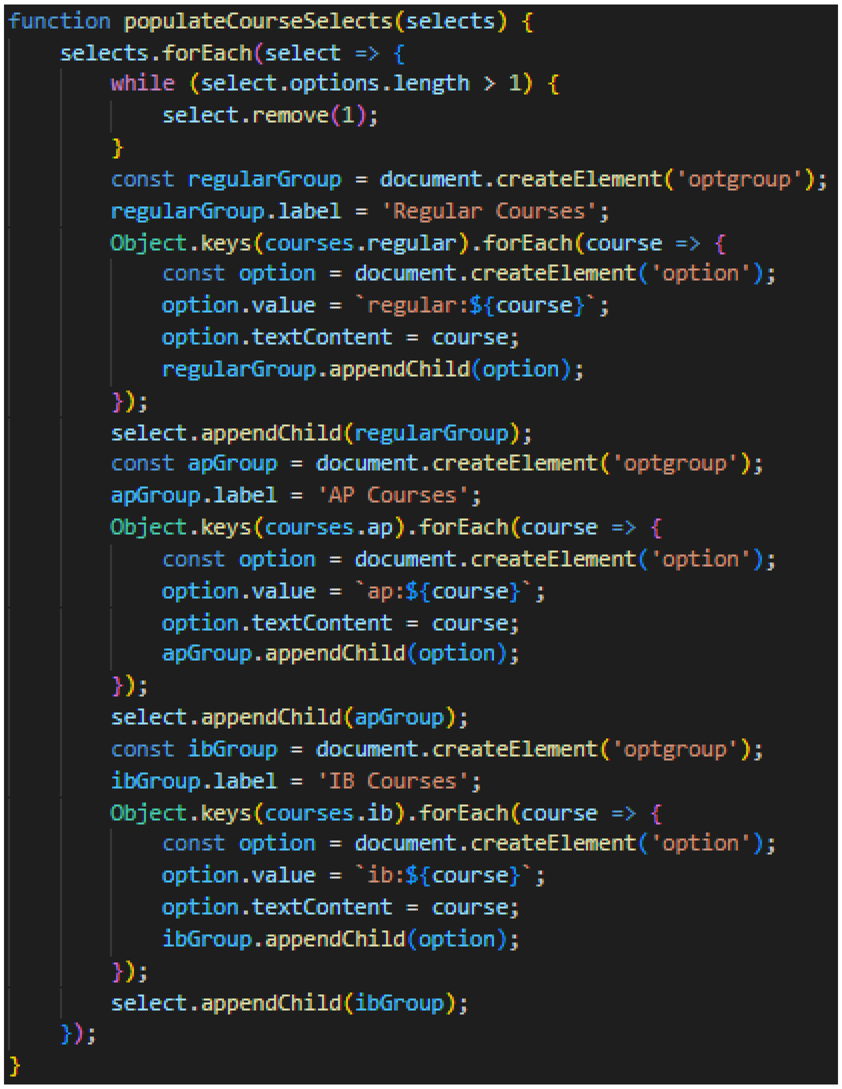
```javascript
function generateSchedule() {
    const currentGPA = parseFloat(document.getElementById('current-gpa').value) || 0;
    const targetGPA = parseFloat(document.getElementById('target-gpa').value) || 0;
    const selectedInterests = Array.from(document.querySelectorAll('.interest-tag.active'))
        .map(tag => tag.dataset.interest);

    if (currentGPA < 2.0 || currentGPA > 5.5 || targetGPA < 2.0 || targetGPA > 5.5) {
        alert('Please enter a valid GPA between 2.0 and 5.5');
        return;
    }

    if (!currentGPA || !targetGPA || selectedInterests.length === 0) {
        alert('Please enter your current GPA, target GPA, and select at least one academic interest.');
        return;
    }
    const gpaGap = targetGPA - currentGPA;
    const recommendedCourses = [];
    const numAdvancedCourses = Math.ceil(Math.abs(gpaGap) / 0.0488) * (gpaGap > 0 ? 1 : -1);
    const availableCourses = {
        advanced: [],
        regular: []
    };

    ['ap', 'ib'].forEach(type => {
        Object.entries(courses[type]).forEach(([name, data]) => {
            if (selectedInterests.includes(data.category)) {
                availableCourses.advanced.push(new Course(name, data.category, type));
            }
        });
    });

    Object.entries(courses.re
        if (selectedInterests
            availableCourses.regular.push(new Course(name, data.category, 'regular'));
        }
    });

    availableCourses.advanced.sort(() => Math.random() - 0.5);
    availableCourses.regular.sort(() => Math.random() - 0.5);

    for (let i = 0; i < 8; i++) {
        if (i < Math.abs(numAdvancedCourses)) {
            const course = availableCourses.advanced[i % availableCourses.advanced.length];
            recommendedCourses.push(course);
        } else {
            const course = availableCourses.regular[i % availableCourses.regular.length];
            recommendedCourses.push(course);
        }
    }

    recommendedCourses.forEach((course, index) => {
        const blockNum = index + 1;
        const courseSelect = document.getElementById(`block${blockNum}-class`);
        const gradeSelect = document.getElementById(`block${blockNum}-grade`);

        courseSelect.value = `${course.type}:${course.name}`;
        gradeSelect.value = 'A';
    });
    calculateGPA();
}
```

Explanation: This function automates the scheduling process, ensuring students meet their academic targets while considering their personal interests. It does all of the following to help make the algorithm more personalized:

- Calculates how many AP/IB courses are needed to reach the target GPA
- Filters courses based on student's academic interests
- Ensures a balanced schedule by mixing regular and advanced courses
- Randomizes course selection within each category to provide variety
- Handles edge cases like insufficient courses in a category using modulo operator

The code is stored within the function generateSchedule() and calls the calculateGPA() function at the end after generating the schedule. The first if statement is used to send the user error handling as part of the success criteria if the GPA the user entered is not within an acceptable range. The second if statement is error handling that checks to make sure that the user completely filled out the form. The forEach function is a loop that helps assign classes into the availableCourses variable to be displayed on the page. The code is suitable because it creates personalized schedules that align with both academic goals and interests while maintaining schedule balance.

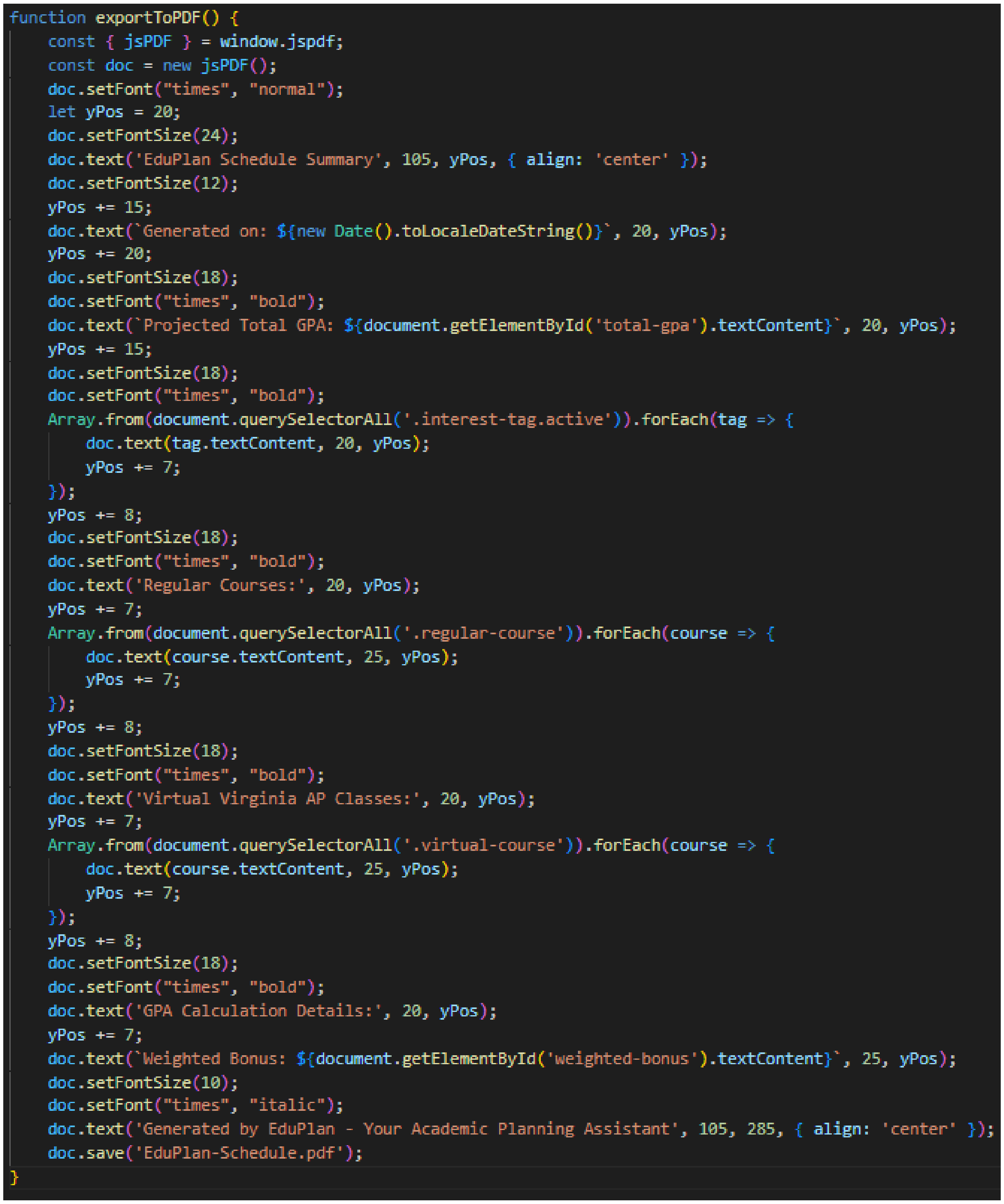
4. Course Selection Function



```javascript
function populateCourseSelects(selects) {
    selects.forEach(select => {
        while (select.options.length > 1) {
            select.remove(1);
        }
        const regularGroup = document.createElement('optgroup');
        regularGroup.label = 'Regular Courses';
        Object.keys(courses.regular).forEach(course => {
            const option = document.createElement('option');
            option.value = `regular:${course}`;
            option.textContent = course;
            regularGroup.appendChild(option);
        });
        select.appendChild(regularGroup);
        const apGroup = document.createElement('optgroup');
        apGroup.label = 'AP Courses';
        Object.keys(courses.ap).forEach(course => {
            const option = document.createElement('option');
            option.value = `ap:${course}`;
            option.textContent = course;
            apGroup.appendChild(option);
        });
        select.appendChild(apGroup);
        const ibGroup = document.createElement('optgroup');
        ibGroup.label = 'IB Courses';
        Object.keys(courses.ib).forEach(course => {
            const option = document.createElement('option');
            option.value = `ib:${course}`;
            option.textContent = course;
            ibGroup.appendChild(option);
        });
        select.appendChild(ibGroup);
    });
}
```

This function dynamically populates course selection elements, enhancing user interaction by providing a responsive and personalized experience. The function populateCourseSelects(selects) takes in the parameter that holds a variable for every single course option. After the computer generates course recommendations, this function populates the drop-down menu that allows users to customize the recommendations by switching the classes for different ones as per success criterion 5.

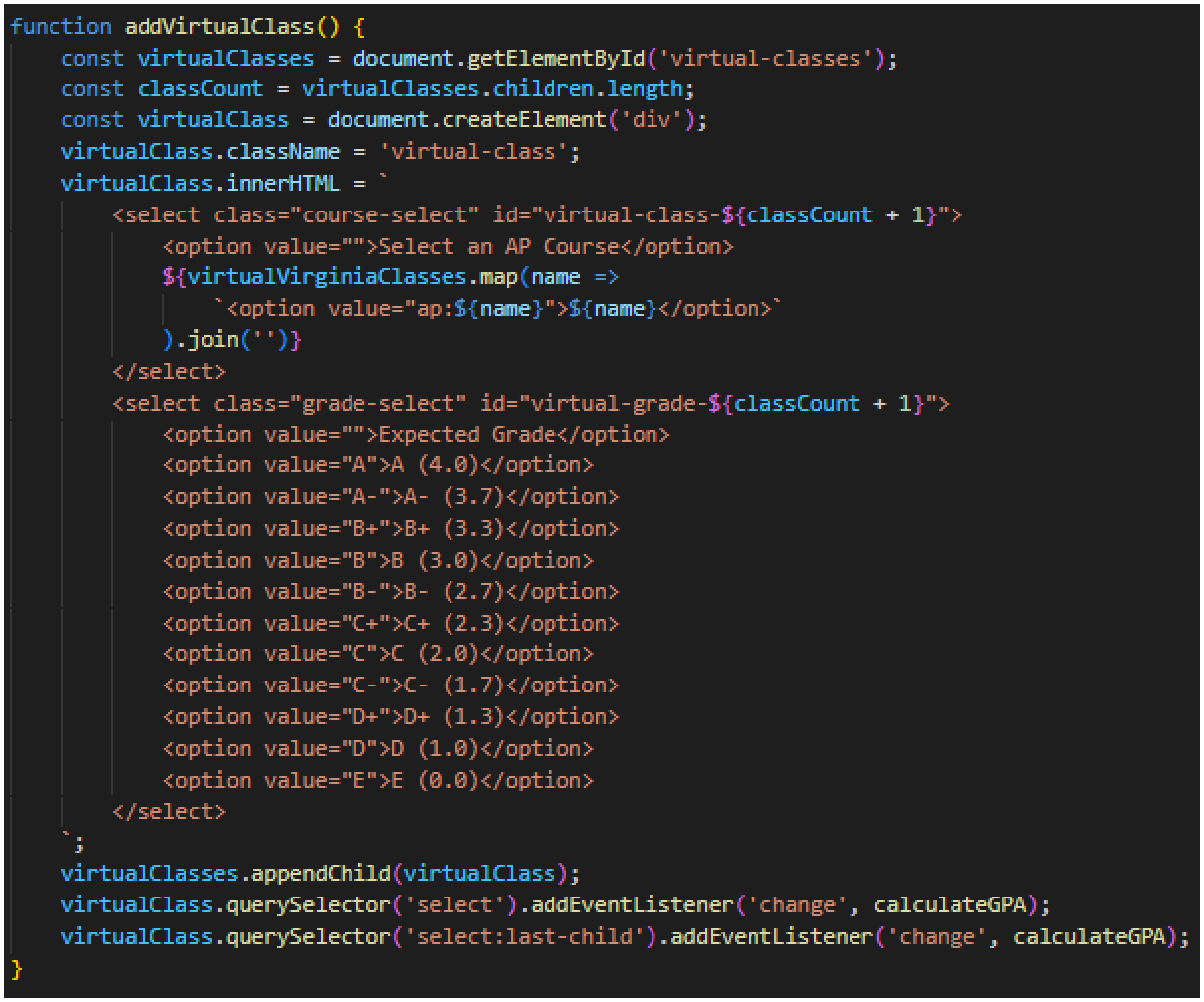
5. Export to PDF Function



```
function exportToPDF() {
    const { jsPDF } = window.jspdf;
    const doc = new jsPDF();
    doc.setFont("times", "normal");
    let yPos = 20;
    doc.setFontSize(24);
    doc.text('EduPlan Schedule Summary', 105, yPos, { align: 'center' });
    doc.setFontSize(12);
    yPos += 15;
    doc.text(`Generated on: ${new Date().toLocaleDateString()}`, 20, yPos);
    yPos += 20;
    doc.setFontSize(18);
    doc.setFont("times", "bold");
    doc.text(`Projected Total GPA: ${document.getElementById('total-gpa').textContent}`, 20, yPos);
    yPos += 15;
    doc.setFontSize(18);
    doc.setFont("times", "bold");
    Array.from(document.querySelectorAll('.interest-tag.active')).forEach(tag => {
        doc.text(tag.textContent, 20, yPos);
        yPos += 7;
    });
    yPos += 8;
    doc.setFontSize(18);
    doc.setFont("times", "bold");
    doc.text('Regular Courses:', 20, yPos);
    yPos += 7;
    Array.from(document.querySelectorAll('.regular-course')).forEach(course => {
        doc.text(course.textContent, 25, yPos);
        yPos += 7;
    });
    yPos += 8;
    doc.setFontSize(18);
    doc.setFont("times", "bold");
    doc.text('Virtual Virginia AP Classes:', 20, yPos);
    yPos += 7;
    Array.from(document.querySelectorAll('.virtual-course')).forEach(course => {
        doc.text(course.textContent, 25, yPos);
        yPos += 7;
    });
    yPos += 8;
    doc.setFontSize(18);
    doc.setFont("times", "bold");
    doc.text('GPA Calculation Details:', 20, yPos);
    yPos += 7;
    doc.text(`Weighted Bonus: ${document.getElementById('weighted-bonus').textContent}`, 25, yPos);
    doc.setFontSize(10);
    doc.setFont("times", "italic");
    doc.text('Generated by EduPlan - Your Academic Planning Assistant', 105, 285, { align: 'center' });
    doc.save('EduPlan-Schedule.pdf');
}
```

Explanation: This function, exportToPDF(), uses the jsPDF library to export the schedule as a PDF, providing a convenient way for users to save and share their schedules. It creates a custom font size for every section of the doc, takes all of the computer generated schedule data, the project GPA, predicted grades for each class, current GPA, weighted GPA bonus, and each course by section onto the form using the document.getElementById function to sort and organize.

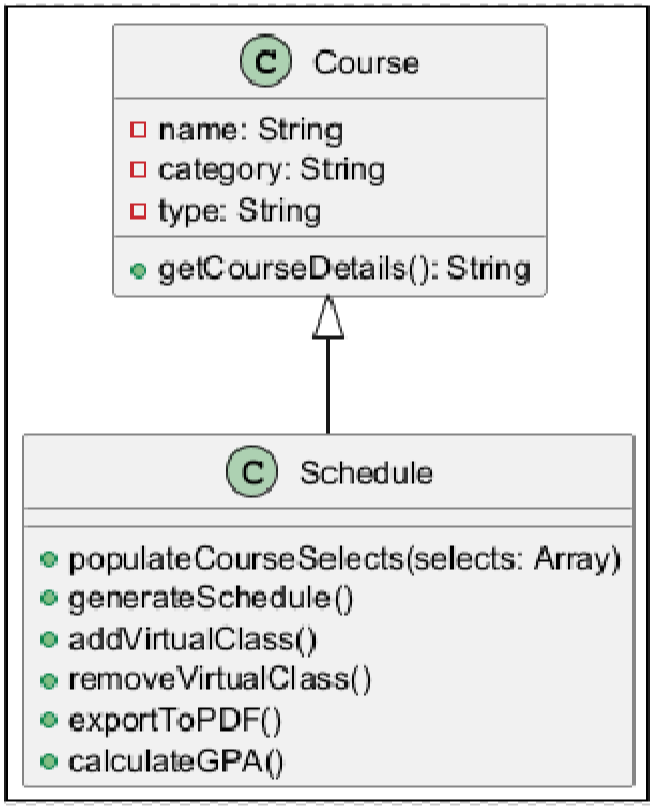
6. addVirtualClass Function



```
function addVirtualClass() {
    const virtualClasses = document.getElementById('virtual-classes');
    const classCount = virtualClasses.children.length;
    const virtualClass = document.createElement('div');
    virtualClass.className = 'virtual-class';
    virtualClass.innerHTML = `
        <select class="course-select" id="virtual-class-${classCount + 1}">
            <option value="">Select an AP Course</option>
            ${virtualVirginiaClasses.map(name =>
                `<option value="ap:${name}">${name}</option>`
            ).join('')}
        </select>
        <select class="grade-select" id="virtual-grade-${classCount + 1}">
            <option value="">Expected Grade</option>
            <option value="A">A (4.0)</option>
            <option value="A-">A- (3.7)</option>
            <option value="B+">B+ (3.3)</option>
            <option value="B">B (3.0)</option>
            <option value="B-">B- (2.7)</option>
            <option value="C+">C+ (2.3)</option>
            <option value="C">C (2.0)</option>
            <option value="C-">C- (1.7)</option>
            <option value="D+">D+ (1.3)</option>
            <option value="D">D (1.0)</option>
            <option value="E">E (0.0)</option>
        </select>
    `;
    virtualClasses.appendChild(virtualClass);
    virtualClass.querySelector('select').addEventListener('change', calculateGPA);
    virtualClass.querySelector('select:last-child').addEventListener('change', calculateGPA);
}
```

Explanation: This function, addVirtualClass(), adds a new virtual class to the schedule, allowing users to include online courses in their academic plan. This function starts with variables that are created from form data about if the student wants to take virtual classes and how many they want. The virtualClass.innerHTML lines creates HTML code that creates each Virtual Virginia class with a dropdown menu for the class options, and a drop down menu for the project grade of each class. It then appends it to the page using .appendChild calls the calculateGPA function to recalculate the new GPA with these new course updates.

UML Diagram:


Course
name: String
category: String
type: String
getCourseDetails(): String
Schedule
populateCourseSelects(selects: Array)
generateSchedule()
addVirtualClass()
removeVirtualClass()
exportToPDF()
calculateGPA()