```
expect.extend({
    toHaveGPA(actual, GPA) {
        const { grades } = actual.data.user;

        const actualGPA =
            Object.keys(grades)
                .map(course => BelAir.computeGpa(grades[course]))
                .reduce((total, grade) => (total += grade)) /
            Object.keys(grades).length;

        const { first, last } = actual.data.user.name;
        const name = `${first} ${last}`;

        const pass = actualGPA >= GPA;

        const message = pass
            ? () => `${name} should not have GPA ${gpa}` // message for .not
            : () => `${name} should have a ${GPA} GPA, not ${actualGPA}`;

        return { actual: actual, message, pass };
    }
});
```

Not Fresh

```
test("will smith", async () => {
    const response = await BelAir.request("will");

    const gpa =
        Object.keys(response.data.user.grades)
            .map(course => BelAir.computeGpa(response.data.user.grades[course]))
            .reduce((total, grade) => (total += grade)) /
        Object.keys(response.data.user.grades).length;

    expect(gpa).toBeGreaterThanOrEqual(2.4);
});
```