```
test("ashley banks", async () => {
    const response = await BelAir.request("ashley")

    response.data.user.grades =
        Object.keys(response.data.user.grades)
            .map(course => BelAir.computeGpa(respon
            .reduce((total, grade) => (total += gra
        Object.keys(response.data.user.grades).leng

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

    expect(response.data.use toMatchSnapshot(
);


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

    const gpa =
        Object.keys(response.data.user.grades)
            .map(course => BelAir.computeGpa(res
            .reduce((total, grade) => (total +=
        Object.keys(response.data.user.grades).l

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

```
import * as BelAir from "../bel-air-academy";

test("uncle phil", async () => {
    const response = await BelAir.request("phil

    expect(response.data.user.name.first).toBe("
    expect(response.data.user.name.last).toBe("P
    expect(response.data.user.role).toBe("parent
});


test("carlton banks", async () => {
    const response = await BelAir.request("carlton
    expect(response.data.user).toEqual({
        id: "carlton",
        role: "student",
        name: { first: "Carlton", last: "Banks" },
        grades: {
            literature: "A",
            math: "A",
            gym: "A",
            health: "A",
            chemestry: "A"
        }
    });
});
```

Not Fresh

```
test("uncle phil", async () => {
    const response = await BelAir.request("philip");

    expect(response).toHaveFullName("Uncle Phil");
    expect(response).toBeParent();
});
```