

Human Care Systems Coding Challenge

Steps to build the integration tool

- Please follow the github repository README for the instructions on how to clone, install, run and test the project (<https://github.com/vrdhoke/Integration-Tool>)
- Created node project using npm init
- Created modules for each use-case and exported to be used in other files e.g app.js.
 - Import CSV data and store in MongoDB “patient” collection , database name is “humancaredata”

```
Modules > JS ImportData.js > [x] ImportData
1  const mongodb = require("mongodb").MongoClient;
2  var url = "mongodb://localhost:27017/";
3  const csvtojson = require("csvtojson");
4  const ImportData = async () => {
5      var result = "";
6
7      const db = await mongodb.connect(url);
8      const dbo = db.db("humancaredata");
9      var csv = [];
10     await csvtojson()
11         .fromFile("Humancare.csv")
12         .then(async (csvData) => {
13             csv = csvData;
14         });
15
16     const patientData = await dbo.collection("patient").insertMany(csv);
17
18     return patientData.insertedCount;
19 };
20
21 exports.ImportData = ImportData;
22
```

- Finding PatientIDs which have missing elements and CONSENT as Y

```
Modules > JS MissingEmails.js > [x] missingEmail
1  const mongodb = require("mongodb").MongoClient;
2  var url = "mongodb://localhost:27017/";
3  const missingEmail = async () => {
4      const db = await mongodb.connect(url);
5      const dbo = db.db("humancaredata");
6      var query = { CONSENT: "Y", "Email Address": "" };
7      const missingEmail = await dbo.collection("patient").find(query).toArray();
8
9      var len = missingEmail.length;
10     var memberId = [];
11     missingEmail.forEach(function (item) {
12         memberId.push(item["Member ID"]);
13     });
14     return memberId;
15 };
16
17 exports.missingEmail = missingEmail;
```

- Finding PatientIDs which have missing First Name

```

Modules > JS MissingFName.js > [x] missingFName
1  const mongodb = require("mongodb").MongoClient;
2  var url = "mongodb://localhost:27017/";
3  const missingFName = async () => {
4      const db = await mongodb.connect(url);
5      const dbo = db.db("humancaredata");
6      var query = { "First Name": "" };
7      const missingFName = await dbo.collection("patient").find(query).toArray();
8
9      var len = missingFName.length;
10     console.log(missingFName);
11     return missingFName;
12 };
13
14 exports.missingFName = missingFName;

```

- Schedule Email and store in “emails” collection

```

Modules > JS SchedulEmail.js > [x] ScheduleEmail
1  const mongodb = require("mongodb").MongoClient;
2  var url = "mongodb://localhost:27017/";
3  const ScheduleEmail = async () => {
4      const db = await mongodb.connect(url);
5      const dbo = db.db("humancaredata");
6      var query = { CONSENT: "Y" };
7      const consentY = await dbo.collection("patient").find(query).toArray();
8      // console.log(consentY);
9
10     var emails = [];
11     consentY.forEach(function (patient) {
12         var memberId = patient["Member ID"];
13         // console.log(memberId);
14         var today = new Date();
15         for (var i = 1; i < 5; i++) {
16             today.setDate(today.getDate() + 1);
17             var email = {
18                 id: memberId + "_" + i,
19                 name: "Day " + i,
20                 scheduled_date: today.toISOString().slice(0, 10),
21                 patientId: memberId,
22             };
23             emails.push(email);
24         }
25     });
26
27     mongodb.connect(url, function (err, db) {
28         if (err) throw err;
29         var dbo = db.db("humancaredata");
30         dbo.collection("emails").insertMany(emails, function (err, res) {
31             if (err) throw err;
32             console.log("Number of documents inserted: " + res.insertedCount);
33             db.close();
34         });
35     });
36
37     return emails;
38 };
39
40 exports.ScheduleEmail = ScheduleEmail;

```

(method) Date.toISOString(): string
Returns a date as a string value in ISO format.

- **Automation Test/Unit Tests using Jest for Nodejs**

I wrote multiple automated/unit test cases to validate the functionality and used Jest automation test framework to write the test cases.

“Integration-tool/Test/humancare.test.js” contains the unit test cases

- Unit test to check if the data is imported to “**patient**” collection without issue

```
describe("Unit Test cases for Human Care System Integration Tool", () => {
  describe("Unit Test import to collection", () => {
    test("Import patient data", async () => {
      var csv = [];
      await csvToJson()
        .fromFile("Humancare.csv")
        .then((csvData) => {
          csv = csvData;
        });
      var len = csv.length;
      expect.assertions(2);
      await ImpData.ImportData().then((data) => {
        expect(data).toBe(len);
      });
      await patientData.ViewPatientCollection().then((data) => {
        expect(data.length).toBe(len);
      });
    });
  });
});
```

- Unit test to check if the imported data matches with csv data or not

```
describe("Unit test Comparing the CSV and collections data", () => {
  test("Compare patient data", async () => {
    var csv = [];
    await csvToJson()
      .fromFile("Humancare.csv")
      .then((csvData) => {
        csv = csvData;
      });

    expect.assertions(1);
    const data = await patientData.ViewPatientCollection();
    var len = csv.length;
    data.forEach(function (item) {
      delete item._id;
    });

    //Comparing the CSV data from the
    await expect(data).toEqual(csv);
  });
});
```

- Unit test to check the missing first name of the patient from “patient” collection

```
describe("Unit Test for missing first names", () => {
  test("Test missing first name of patient", async () => {
    var csv = [];
    await csvtojson()
      .fromFile("Humancare.csv")
      .then((csvData) => {
        csv = csvData;
      });
    var count = 0;
    csv.forEach(function (item) {
      if (item["First Name"] == "") {
        count += 1;
      }
    });
    expect.assertions(1);
    const data = await findPatient.missingFName();
    var len = csv.length;
    await expect(data.length).toBe(count);
  });
});
```

- Unit test to check the missing email address

```
describe("Unit Test missing email addresses with CONSENT Y", () => {
  test("Test missing email addresses of patient", async () => {
    var csv = [];
    await csvtojson()
      .fromFile("Humancare.csv")
      .then((csvData) => {
        csv = csvData;
      });
    var count = 0;
    var patientIds = [];
    csv.forEach(function (item) {
      if (item.CONSENT == "Y" && item["Email Address"] == "") {
        patientIds.push(item["Member ID"]);
      }
    });
    expect.assertions(1);
    const data = await missingEmail.missingEmail();
    await expect(data).toEqual(patientIds);
  });
});
```

- Unit test to validate the scheduled email on Day1,Day2,Day3 and Day4

```
describe("Unit test for Schedule Emails", () => {
  test("Schedule Emails", async () => {
    var csv = [];
    await csvToJson()
      .fromFile("Humancare.csv")
      .then((csvData) => {
        csv = csvData;
      });
    var count = 0;
    csv.forEach(function (item) {
      if (item.CONSENT == "Y") {
        count += 1;
      }
    });
    var len = csv.length;
    expect.assertions(1);
    await ScheduleEmail.ScheduleEmail().then((data) => {
      //Each patient has 4 emails scheduled for Day1,Day2,Day3,Day4
      expect(data.length).toBe(count * 4);
    });
  });
});
```

- Dropping the “Patient” and “Emails” collection at the end of the test suit so that the test cases can rerun without any issue

```
describe("Drop Patient data every time to rerun the test cases", () => {
  it("Drop patient data", async () => {
    // expect.assertions(1);
    return await DropCSVData.DropCSVCollection().then((data) => {
      expect(data).toBe(true);
    });
  });
});

describe("Drop Email data every time to rerun the test cases", () => {
  it("Drop Email data", async () => {
    // expect.assertions(1);
    return await DropEmailData.DropEmailCollection().then((data) => {
      expect(data).toBe(true);
    });
  });
});
```

- Result of the Unit Tests (npm run test)

```
PASS Test/humancare.test.js
  Unit Test cases for Human Care System Integration Tool
    Unit Test import to collection
      ✓ Import patient data (255 ms)
    Unit test Comparing the CSV and collections data
      ✓ Compare patient data (9 ms)
    Unit Test for missing first names
      ✓ Test missing first name of patient (30 ms)
    Unit Test missing email addresses with CONSENT Y
      ✓ Test missing email addresses of patient (5 ms)
    Unit test for Schedule Emails
      ✓ Schedule Emails (5 ms)
    Drop Patient data every time to rerun the test cases
      ✓ Drop patient data (266 ms)
    Drop Email data every time to rerun the test cases
      ✓ Drop Email data (54 ms)

Test Suites: 1 passed, 1 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        3.567 s
Ran all test suites.
Force exiting Jest: Have you considered using `--detectOpenHandles` to detect async operations
(base) vaibhavdhoke@Vaibhavs-MacBook-Pro ~/Documents/HumanCare/Integration-Tool [main]
```