Ryan Reynolds

Web API Build and Test Instructions

* Development Environment:
  + OS: Windows 10
  + Programming Language: C#
  + Framework: Asp.net Web Api
    - Not compatible with mac
  + IDE: Visual Studio 2017
* Test Environment:
  + Google Chrome Version 75.0.3770.100
  + Postman 6.7.4

1. Build Instructions
   1. Open Reynolds\_API.sln in Visual Studio
   2. Run the project
   3. Running the project in debug mode or releasing it will serve the api on localhost:62851
   4. The API can be consumed in the browser via the web application or the endpoints can be consumed via an API dev tool, such as postman.
      1. The web app has documentation on all of the endpoints and method calls
2. Testing
   1. Testing in the web application:
      1. All data is displayed in the index html upon submission of the forms
      2. Informal testing can be performed in the web application
      3. Input & Analysis
         1. files uploaded are saved under Reynolds\_API/Reynolds\_API/Files along with a json storing the analytics calculated by the api.
         2. The path to the most recent upload is cached locally in the Files directory
      4. Scanning
         1. Requires the full path to the directory in order to work.
   2. Unit Testing
      1. All unit test files are located under Reynolds\_API/UniteTestAPI/TestFiles
         1. Note: The paths need to be updated to the local file system.
      2. Any additional formal unit testing can be added to the UnitTestInputAnalysis.cs and UnitTestScanning.cs files
         1. Each exposed method has at least a single unit test already implemented.
         2. Input Analysis: Unit tests read in a ground truth File Model json object. Located in the TestFiles/InputAnalysis directory
         3. Scanning: Unit tests have their ground truth manually entered in.
      3. Unit tests can be run from Visual Studio by building the UnitTestAPI then running the tests in Test Explorer (Test Tool bar-> Run -> All Tests)
3. General Project Structure (MVC Architecture)
   1. Reynolds\_API
      1. Controllers
         1. InputAnalysis
            1. Controller for the Input & Analysis challenge
         2. Scanning
            1. Controller for the Scanning challenge
      2. Models
         1. FileModel
            1. Contains the structure for the Input Analysis Data
         2. DirectoryModel
            1. Contains the structure for the Scanning Data
      3. Views
         1. InputAnalysis
            1. Index

Gui for consuming the api in a browser

* + - * 1. Documentation

In depth method and endpoint descriptions

* + - 1. Scanning
         1. Index

Gui for consuming the api in a browser

* + - * 1. Documentation

In depth method and endpoint descriptions

* 1. UnitTestAPI
     1. UnitTestInputAnalysis
        1. Contains unit tests for each of the exposed endpoints for the input analysis controller
     2. UnitTestScanning
        1. Contains unit tests for each of the exposed endpoints for the Scanning controller