Inventory Exercise Documentation

I. Materials
   a. ASP.net C#
   b. WebAPI/MVC
   c. Postman
   d. GitHub https://github.com/mic615/Valant_Inventory

II. Schema
   a. Label (int)
      i. unique identifier for each item in the inventory.
   b. Type (string)
      i. descriptor of what an object is (ex. “orange”, or “ taco shells”)
   c. IsExpired (bool)
      i. Boolean value that denotes expiration of an object
   d. Expiration (Date) (not actually implemented)
      i. I would include this value if a database were used so that a user could specify an expiration date. (see section VIII.a.ii)

III. Data storage
   a. Storage
      i. as the instructions specified, no database was needed. this API is storing the data state using a text file and file I/O.
      ii. This is a very poor solution for reasons described in section VIII.a

   b. Interfacing
      i. Due to the nature of storage used for this project data is manipulated and gathered using a list of Item objects
      ii. The Data is updated using a list of csv style strings representing rows of data (see IX.a)

IV. RESTful Methods

API	Description
GET api/Items	Get all items in inventory
GET api/Items/{label}	Get an item from inventory based on unique label
POST api/Items	Add an Item to the inventory
PUT api/Items/{label}	Update an item based on request's body content currently only modifies IsExpired as there are no other modifiable values in the current schema
DELETE api/Items/{label}	Remove an item from the inventory

V. Notifications and Logging

a. **Notifications are sent in the responses of every request including those specified in the requirements.**

b. **Notifications sent out to the console for your convenience however they should be removed or commented out in a production release.**

VI. Security

a. **The major security flaw in this API is the use of a text file for data storage as illustrated in section VIII.a**

VII. Tests

More tests need to be added to ensure full branch coverage however, the following test cases are implemented via unit testing:
Gettems()
GetItem(label)
Post good item (pass)
Post bad item (fail)

I would like to implement
ExpireItem(Label, item) pass

UnexpireItem(Label, item) Fail
DeleteItem(Label) pass

VIII. Limitations

a. Using a text file for storage
    i. Data integrity assurances are all handled programmatically. There is no second line of defense (i.e label auto incrementing, default values, or "not null" protection)
    ii. We lose access to valuable features provided by a database such as using a database trigger to automatically update IsExpired on an expiration date.
    iii. Inefficient non-scalable storage method
    iv. Lack protected from outside data manipulation which could corrupt data and cause data loss
    v. No connection string / database authentication required
    vi. Harder to manage user authentication and user table would basically be another text file

IX. Improvements

a. Most of the limitations in section VIII are removed by simply adding database architecture to the solution.
b. I would then add a user table for token based authentication and accountability in logging (i.e what user changed this value)
c. If using a database, I would use a DBEntity object using entity frameworks this uses a connection string to authenticate to the DB to make any changes or access data.
d. I would like to add Log4net implementation

X. Build instructions

a. Open project's folder and run inventory.sln
b. Click the test button and press run all tests
c. Run the application
d. Send your request using your preferred method (I use postman)