Title: Meter Reading API

(any assumptions I have made i have marked with PO Confirmed, to show that this would be a decision from the Product Owner)

Story Points: 5 (medium complexity)

Problem statement:

As an Energy Company Account Manager,
I want to be able to load a CSV file of Customer Meter Readings
So that We can monitor their energy consumption and charge them accordingly

Acceptance Criteria:

- The url must be /meter-reading-uploads
- The endpoint must be able to process a CSV
- Valid entries must be stored in a database
- The response must contain the total of successful and failed readings
- Validation:
    - Duplicate entries are not allowed
    - The reading must have a valid AccountId
    - Reading must be numerical to 5 places NNNNN
    - Negative values are invalid

Technical details:

Csv Processing

- Additional columns should be ignored and not cause errors *(found in test data - PO confirmed)*
- The original meter reading value should be preserved and then parsed to conform to the validation rules
- CsvHelper has been identified as a library to process the CSV files for us

Validation

- The test data contains a considerable number of values that are too short and do not conform to the required format, these will be considered invalid *(PO confirmed)*
- The meter reading has a potential to loop back to zero, this was raised by the team and considered out of scope for this initial development *(PO confirmed)*

Database design

- There is potential for a foreign key constraint on accountId, the team made the decision not to tightly couple the tables at this point

Test Cases:

Given a POST request
And the csv is not present
Then return bad request

Given a POST request
And the csv is present
When the data is proceeded
And all data is correct
Then the success count equals the uploaded count

Given a POST request
And the csv is present
When the data is proceeded
And all data is incorrect
Then the failure count equals the uploaded count

Given a POST request
And the csv is present
When the data is proceeded
And some data is correct
Then the failure count equals the uploaded count

Tasks:

Bootstrapping

- [x] ~~B1: Create skeleton Project~~
  - [x] ~~Build project~~
  - [x] ~~Add docker orchestration~~
  - [x] ~~Check project loads~~
  - [x] ~~Add testing project~~
- [x] ~~Research and select database engine (assuming Sql Server)~~
  - [x] ~~Install ef core~~
  - [x] ~~Deploy database to docker-compose~~
  - [ ] Connect successful to the Db

Account Data

- [ ] A1 Create database
- [x] ~~A2 Create database seeding~~
- [x] ~~A3 Create Accounts Repository~~

Meter reading Api

- [x] ~~M1 Create minimal Api~~
- [x] ~~M2 Create Meter Reading Service~~
- [ ] M3 Create Validator
    - [x] ~~Create basic validator skeleton~~
    - [x] ~~Create accountId validation~~
    - [x] ~~Create meter reading validation~~
    - [ ] Make date of meter reading validation
- [ ] M4 Create database context

Notes:
Database setup took longer than expected.

Didn't complete:
Unit of work for database saving
Returning and converting the valid collection of meter readings to the database
Adding database rollbacks

I chose to use the minimal API style because the brief was short and I knew there would be no further development, in production, I would consider and lean towards using classic controllers.