design

1 ContentServer

- read data from a txt file
- sent HTTP PUT request to aggregation server
- check whether data is valid
- print response from aggregation server
- read the hostname:port and filepath

2 AggregationServer

- handle requests
  - if "GET" then send the weather data if server has not recieved weather data from content server, then give relavent response
  - if "PUT" then accept the data and store it in a local json object
- give restful response
  - first time get weather data -- 201
  - update weather data -- 200
  - empty weather data -- 205
  - requests other than "GET" and "PUT" -- 400

3 GetClient

- send GET request to Aggregation Server
- print the weather data from server
- if needed, user can get data again by input specified word(in my project, user can input "get" to get and print the weather data)
- die in 30 seconds if user does not do any operation
- read the port number from user
  - if wrong port, then print error and ask user to input again

4 other details

- communication
  - all use socket to communicate and the request is in HTTP format
- data format
  - all use json format

Presented with xmind