REST API and Interface

=> Load the data (in the accompanying file) into a datastore and analyse it

=> Use a programming language of your choice to build an API layer on top of this to accomplish the following tasks:

- Enable filter on category, brand, source, subcategory
- Search on title, sku
- List out all the products with source as ‘Amazon’

=> Build a browsable interface using HTML/CSS/JS for the data set provided. To complete the following tasks

- Search products based on keywords
- Ability to apply filters (price range, category, product features, stock status)
- Pagination of results

=> Make sure the code is production ready. Extra credits to optimize the system at various components and their intersections. Analyse the system and you’ll find many such places. Your task is to make it stable and fast.

=> Feel free to try out extra things which you believe can improve the system in anyway and most importantly enjoy the task!

Instructions

- Please use a relational datastore for storing data
- Please refrain from using an ORM
- Please make sure that the API responses are in standard and readable format
- Please make sure to construct both the Backend and Frontend to start with and then add more APIs to it.
- Write a small doc that explains the data model and design decisions.
- Your code should be deployable on any standard Linux distro with minimal intervention. Share a setup script and deployment instructions along with the rest of the code.

Design Problem: DWRedis

Design an in memory <key, value> store such as Redis. Let's call it DWRedis.

Requirements:
- It should support the major Redis operations such as SET, GET, and SAVE.
- There should be a service that supports start, restart, and stop of DWRedis.
- When the service is started it should load data from a file at a standard location.
- When the service is stopped, it should dump data to the above file.

Some questions you need to consider
- How are you going to implement TTL?
- How will you manage a keyspace that is larger than the memory allocated for DWRedis?
- How will you implement versioning and checkpointing?
- How are you going to handle crash recovery?
- Assume that you can run your DWRedis in a clustered mode. If a new node is added, how are you going to redistribute the keyspace?
- How fast is your DWRedis going to be? Can you estimate the throughput--with proper reasoning--without implementing and testing this?

What you need to deliver?
- A design doc that has the following:
    - choice of data structures
    - design details for each operation
    - schematics for different workflows (load, dump, checkpointing, crash recovery)
    - design considerations to ensure performance, reliability, correctness, and scalability
- Answers to the questions in the previous section.