Overview

Create a restaurant application which accepts menu items from various tablets in the restaurant. This application must then store the item along with a countdown for the item to be ready to serve. The application must be able to give a quick snapshot of any or all items on its list at any time.

The focus will be on data structure choice, API design and implementation, internal implementation (especially regarding data correctness, multi-threaded capacity, and proper unit testing) and avoiding side effects. Please use Scala as the programming language and the Cats library to help writing pure functional code.

Please keep everything simple and focus on making each piece as solid as possible. The parts that are in the solution should be tested. If time runs out, it is preferable to have a few pieces that are well-written and functional with unit tests rather than a broader solution which is buggy and untestable.

Important Notes:

1) Please regard this as a chance to showcase your ability to pick up a new language and programming concept, understand its basics, and apply that understanding in a short amount of time. Picking a new or rather unfamiliar language and showing that you can write some basic code using the strengths of that language, even if the code isn't mastery-level and doesn't include super high-powered features, is a powerful showcase of your software engineering skills.

2) The solution should be as clean as you can make it (deployable, tested, documented, etc.), however, please bear in mind that the point of this is to showcase your skills, particularly with functional programming, API design, data structure choice, and data manipulation techniques. The desire to use third-party tools to perform data updating, table and database management, etc. which would ordinarily be in a production-ready design is much appreciated and understood, however, these tools hide your skills behind already-built libraries, and make it difficult for us to understand your level with regards to skills vital to success at Paidy.

3) Examples of Scala libraries which are okay to use:
    - http4s
    - cats (pure functional programming library)
    - Fs2 or Monix (pure functional streaming libraries)
    - Anything in standard scala library
    - Any library that is part of the typelevel ecosystem

Good luck!

Submission

You can either submit your work with a GitHub repository or send us a zip file with your source files. In both cases, please provide a README file to indicate how to build and run your source as well as the expected outputs.

System Actors

The application - Running on a "server" and accepting calls from devices carried by restaurant staff to process guest's menu orders. This is where the bulk of time should be spent.

The client - Multiple tablets carried by restaurant staff to take orders. These will send requests to the "server" to add, remove, and query menu items for each table. Please make this as simple as possible.

Requirements

1. The client (the restaurant staff "devices" making the requests) MUST be able to: add one or more items with a table number, remove an item for a table, and query the items still remaining for a table.
2. The application MUST, upon creation request, store the item, the table number, and how long the item will take to cook.
3. The application MUST, upon deletion request, remove the item for the table number.
4. The application MUST, upon query request, show all items for the requested table number.
5. The application MUST accept at least 10 simultaneous incoming add/remove/query requests.
6. The client MAY assign a table number between 1-100 (but the same table number must always refer to the same table).
7. The application MAY assign a length of time for the item to prepare as a random time between 5-15 minutes.
8. The application MAY keep the length of time for the item to prepare static (in other words, the time does not have to be counted down in real time, only upon item creation and then removed with the item upon item deletion).

Allowed Assumptions

- The time to prepare does not have to be kept up-to-date. It can also just be generated as some random amount of time between 5 and 15 minutes.
- The table number can be any number, but it has to be consistent. So if a request comes in for table 4, any other requests for table 4 refers to the same table.
- "Clients" can be simulated as a simple stream / background task in the main function, calling the main server application with a variety of requests. There should be more than one, preferably around 5-10 running at any one time.

- The API is up to the developer. HTTP REST is acceptable, but direct API calls are also acceptable if they mimic an HTTP REST-like API (e.g. *api_call1(string id, string resource)*, etc.).