What is it.
This is an Android App emulating a server and a client. The client is self explanatory, the server is emulated by a service. All code is written in Kotlin. All threading is handled via coroutines.

Channels
orderUpdateChannel -- exported by the ShelfManager class, it emits once a second. It is used to age all items on a shelf
shelfStatusChannel -- exported by the ShelfManager class, it emits in response to an order arival from the service
heartBeatChannel -- exported by the Service, it emits once a second. This value is monitored by the ShelfManager
driverArrivalChannel -- exported by the Service, it emits at random intervals between 2 to 8 seconds. This value is monitored by the ShelfManager
orderNotificationChannel -- exported by the Service, it emits at random intervals based on a Poisson distribution. This value monitored by the ShelfManager.

The MainActivity, does not interact with the Service. It only interacts with the ShelfManager. The code in the MainActivity is responsible for updating U.I. and managing the life cycle.

The ShelfManager does all the heavy lifting of maintaining various counts and updating orders. All communication is asynchronous.

Organization
The source code resides in various subdirectories. Each subdirectory is named with a moniker descriptive of the contents of that directory. I did not provide any Unit testing.

Running the Application
To run the application you will need a laptop with Android Studio installed. The App is written for a tablet in landscape mode only. There was too much data being displayed to see orders on a phone.

Building the Application
To build it, you will need access to Android Studio. Open the included project and select “Build” from the menu. If you have problems building it, please let me know.

How it is modeled
The kitchen model consists of four shelves. Each shelf is modeled by a recycler view. Each recycler view contains a detail item that represents a single orders. The order is displayed as a card.

Service

An Android in-proc Service is used to model a remote Server. This service contains two threads. One uses random values between 2 and 8 inclusive to dispatch drivers to pick up the orders. The other thread simulates order arrival using Poisson Distribution.

Data
The JSON data provided has been added to a resource file and compiled into the app.

Order Selection
The Order thread sleeps for N seconds where N is generated to simulate Poisson Distribution. Each time the thread wakes up, it selects a random order from the JSON array and notifies whomever is listening.

Driver Selection
The system does not simulate drivers. What it does is use a random value between 2 and 8 to remove a random order from a shelf.

Loging
The time interval and an running average are written to the system log. To see it, run the application in the emulator or on device and issue a logcat command from a terminal:
adb logcat | grep "=-=-=-=-=" This will filter all messages specific to this project. As an aside, if you monitor the logs long enough, you will notice that the average order delivery time converges to 3.25.

Classes
1. MainActivity
2. RecyclerViewAdapter
3. Constants
4. Various data providers
5. ShelfManager
6. FoodOrderService
7. Shelf classes: BaseShelf, ShelfCold, ShelfFrozen, ShelfHot, ShelfOverflow
8. Utilities