Specification Document

Data that Management Team Receives from Front-Of-House Team

To ensure that we (the management team) can perform all of our tasks, we require the front-of-house team to send us the following data.

An individual table implies a logical grouping of data. Each entry in the table has the proposed data name, the proposed data type and the details of the data. Any information required to interpret the table has been specified directly before the table. In addition, the source (the document or lecture recording from which we have identified this requirement) and the source details (any relevant notes from the source) have been specified directly after the table.

The management team requires that the front-of-house team send us a record of all sales (presumably daily or weekly - the exact frequency is not required for interpretation of the table). The record of sales will be an ArrayList of Sale objects. Each sale will be an object of class Sale. The sale object must contain a saleID (int) field, a total (int) field, a method (String) field and a dishList (ArrayList<String>) field. In order for us to use the saleList as intended, we require the sale object to conform to the requirements detailed below. However, when we request data, we will only request the saleList itself, not the individual instances of data it contains.

Data Name	Data Type	Data Details
saleList	**ArrayList<Sale>**	An ArrayList of Sale objects. The admin team receives a list of sales.
sale	Object	Each sale is an object of class Sale. The sale object must contain a saleID (int) field, a total (int) field, a method (String) field and a dishList (ArrayList<String>) field.
saleID	int	The primary key for a sale.
total	int	The total amount of payment taken.
method	String	The method of payment ("card" or "cash").
dishList	ArrayList<String>	NOTE: name refers to the name of a dish. An ArrayList of name values. We use an ArrayList instead of a HashSet to allow duplicate values. Instead of having a HashMap with name as the key and quantity as the value (which would add complexity), we opt for an ArrayList that only stores the name value. Multiple instances of the same name translates to a quantity greater than 1.
name	String	The name of the dish. We use name as the primary key for a dish (two dishes cannot have the same name). Therefore, we only need name.

Source(s): Case Study -> Front of House -> Billing and Payment

Source details: The case study document states the front-of-house team keeps a record of all sales so that they know which dishes sold well and the monies taken.

The management team requires that the front-of-house team send us a record of all bookings (presumably daily or weekly - the exact frequency is not required for interpretation of the table). The record of sales will be a HashSet of Booking objects. Each booking will be an object of class Booking. The booking object must contain a bookingD (int) field, a name (String) field, a number (String) field and a type (String) field. In order for us to use the bookingSet as intended, we require the booking object to conform to the requirements detailed below. However, when we request data, we will only request the bookingSet itself, not the individual instances of data it contains.

Data Name	Data Type	Data Details
bookingSet	**HashSet<Booking>**	A HashSet of Booking objects.
booking	Object	Each booking is an object of class Booking. The booking object must contain a bookingD (int) field, a name (String) field, a number (String) field and a type (String) field.
name	String	The name the booking is under.
number	String	The telephone number given for the booking.
type	String	The type of booking ("phone", "in-person", "walk-in", "online)

Source(s): Case Study -> Management (-> Tracking)

Source details: The case study document states the admin team keeps track of bookings. Furthermore, under the "Tracking" heading, it is implied/stated that this information is used to predict future bookings.