Order Tracking Application
Application Description and API Interface Points

1. Application summary

   This application creates and records orders of items by customers, and stores the orders, items and customer information in a relational database. Interactions with the application are through an API using JSON scripts.

   The following tables are created and utilized by this application:
   Customer, Ticket, Item and join table Item_Ticket.

2. Description of Table Objects and Data Specifics

2.1 CUSTOMER

Each unique customer can have many orders (tickets), with the same or different items within the order. Customer information is retained if a ticket is deleted. It is possible to modify customer information.

The customer record must be created before an order can be assigned. Customer information required through the API for the order is as follows:

- Customer name (string)
- Customer address (string - Street address)
- Customer phone (string, no format restrictions)
- Customer email (unique string, no format restrictions)

2.1.1 API Access information and available table operations

API access path is: /order_tracking/customer

Supported API Operations

POST: /order_tracking/customer
PUT: /order_tracking/customer/*customerId*
GET: /order_tracking/customer/*customerId*

2.1.2 Sample JSON script for Customer Object is as follows:

```
{
        "customerName": "CUSTOMER NAME",
        "customerAddress": "CUSTOMER ADDRESS",
        "customerPhone": "(XXX) XXX-XXXX",
        "customerEmail": "EMAIL@SOMEDOMAIN.COM"
}
```

2.2 TICKET

Each unique order is referred to as a ticket, because "order" is a reserved keyword. Each unique Ticket can have as many items as desired and is assigned to a single Customer. When the order (ticket) is deleted, the associated items are removed from that table. Tickets can be modified or deleted through the API.

A Ticket must be created and assigned to a Customer before an Item can be added. Ticket information required through the API is as follows:

- Ticket Date (string – desired format is: YYYY-MM-DD)
- Ticket Due Date (string – desired format is: YYYY-MM-DD)
- Ticket Shipment Date (string – desired format is: YYYY-MM-DD)

2.2.1 API access path is: /order_tracking/ticket

Supported API Operations

POST: /order_tracking/customer/*customerId*/ticket
PUT:/ order_tracking/ticket/*ticketId*
GET: /order_tracking/ticket/customer/*customerId*
GET: /order_tracking/ticket/*ticketId*
DELETE: /order_tracking/ticket/*ticketId*

2.2.2 Sample JSON script for Ticket Object is as follows:

```
{
        "ticketDate": "YYYY-MM-DD",
        "ticketDueDate": "YYYY-MM-DD",
        "ticketShipmentDate": "YYYY-MM-DD"
}
```

2.3 ITEM

Each unique Item can be added to as many orders as is required. Items are created through the API as they are assigned to an order. This is a limitation that should be addressed in a future release. Item information required through the API is as follows:

- Item name (string)
- Item price (two place decimal)
- Item UPC (thirteen-digit number) – *number is unique and will not permit duplicates*

2.3.1 API access path is: /order_tracking/ticket/*ticketId*

Supported API Operations

POST: /order_tracking/*ticketId*/*itemId*
POST:/order_tracking/item

2.3.2 Sample JSON script for Item Object is as follows:

```
{
        "itemName" : "ITEM DESCRIPTION",
        "itemPrice" : "XXXX.XX",
        "itemUpc" : "XXXXXXXXXXXXX"
}
```

3. Error Handling

Errors are standard HTTPs error codes, such as 201, 400, 500.