Matthew Toro
onid: torom
CS 496 Mobile and Cloud Software Development
Assignment: Final Project
Due Date: 3/18/18

I did the Hybrid mobile and cloud project for my project. I made a simple REST API backend modeling a store selling board games to customers. It has the following two entities:
Boardgame:
Title (String)
Description (String)
Stock (Int)
Price (Int)

Customer:
FirstName (String)
LastName (String)
Money (Int)
Capacity (Int) (how many games they can carry out of the store)
Inventory (String[])

The relationship between the two entities is a customer can purchase a game or they can return a game they purchased. You can see what games a customer has in their inventory; a game purchased will be added to their inventory, while a game returned will be removed from their inventory. Inventory stores board games as links to their respective pages.

API Main Page:
silicon-perigee-191721.appspot.com

POST /boardgame
- Creates a new boardgame
- Set Header content-type to application/json
- Send request body as JSON and modeled:

```
{
    "title": "test_game",
    "description": "this is a test game",
    "price": 1,
    "stock": 10
}
```

where 'title' and 'description' are strings and 'price' and 'stock' are integers. An id for the boardgame will automatically be generated.
- Requests not modeled like above will be rejected and send a 400 status code
- If successful, returns status code 201

GET /boardgame
- Returns every boardgame in the database

GET /boardgame/{boardgame_id}
- Returns the boardgame with the given id
- If the id is not valid, then it will send a 400 status code back

PATCH /boardgame/{boardgame_id}

- Modifies the boardgame with the given id
- If the id is not valid, then it will send a 400 status code back
- Set Header content-type to application/json
- The request body must be JSON and modeled:

```
{
    "title": "edit_game",
    "description": "this is an edit game",
    "price": 2,
    "stock": 10
}
```

- If the request does not model the above, it will be rejected and send a 400 status code back
- If you do not want to edit a specific item, then send its current value

PUT /boardgame/{boardgame_id}
- Puts the specified boardgame into the customer's inventory
- If the id is invalid, then it will send a 400 status code
- Set Header content-type to application/json
- Send in the body JSON data modeled:

```
{
    "id": "{id string}"
}
```

where 'id' is the string representation of the customer's id
- If the body does not match the above, then it will be rejected and send a 400 status code
- If the boardgame is out of stock or the customer does not have room for it or they do not have the money for it, then it will send a 403 status code
- Upon success, boardgame stock will be decremented, customer's money will be decremented by boardgame price, and the boardgame will be added to customer's inventory via a link

DELETE /boardgame/{boardgame_id}
- Deletes the boardgame with the given id
- If the id is not valid, then it will send a 400 status code back

POST /customer
- Creates a new Customer
- Set Header content-type to application/json
- Send request body as JSON and modeled:

```
{
    "firstName": "John",
    "lastName": "Doe",
    "money": 10,
    "capacity": 1
}
```

where 'firstName' and 'lastName' are strings and 'money and 'capacity' are integers. An id for the customer will automatically be generated.
- Requests not modeled like above will be rejected and send a 400 status code

GET /customer

- Returns every customer in the database

GET /customer/{customer_id}
- Returns the customer with the given id
- If the id is not valid, then it will send a 400 status code back

DELETE /customer/{customer_id}
- Deletes the customer with the given id
- If the id is not valid, then it will send a 400 status code back

PATCH /customer/{customer_id}
- Modifies the customer with the given id
- If the id is not valid, then it will send a 400 status code back
- Set Header content-type to application/json
- The request body must be JSON and modeled:

```
{
    "firstName": "Edit",
    "lastName": "Doe",
    "money": 10,
    "capacity": 1
}
```

- If the request does not model the above, it will be rejected and send a 400 status code back
- If you do not want to edit a specific item, then send its current value

PUT /customer/{customer_id}
- Returns the specified boardgame back to the store for a refund
- If the customer id is not valid, then it will send a 400 status code back
- Set Header content-type to application/json
- The request body must be JSON and modeled:

```
{
    "id": "{id string}"
}
```

where 'id' is the id of the boardgame to be returned.
- If the board game id is not valid, then it will send a 400 status code back
- Upon success, the boardgame will be removed from the customer's inventory and their money will be refunded and the boardgame's stock will increase by 1