Phase I: MyKart eCommerce System

Summary

MyKart is a system that allows users to sell products online. This document outlines the feature deliverable for Phase I of the product. Phase I of this product will be the delivery of components to support Administrator management and Product management.

MyKart Components

API Service

A RESTful backend microservice that supports CRUD operations for resources such as customer, order, product, etc. Returned response payload will be in JSON format. Request payload will be in JSON format. Later phases will require security around every call to this service.

Phase I will support admin management operations and Product Management.

Admin Management

Endpoint to support create new admin user.
Endpoint to support update existing admin user.
Endpoint to support soft-delete of existing admin user.

Admin Properties:
Stores information about an admin. This should include id, password, name, active indicator, created date, last updated date, etc. All Date values are to be stored as String in GMT format (YYYY-MM-DDTHH:MM:SSZ).

Product Management

Endpoint to support getting all products
Endpoint to support getting a specific product for a given product id
Endpoint to support getting a listing of products based on some search filter(s) such as names, categories, etc.
Endpoint to support deleting a product.
Endpoint to support updating a product.

Product Model:

Stores product information. A product will have an id, name, description, unitPrice, and on hands amount. Additionally each product will store the date the product was added to the system, the id of the administrator and added the product, the date the product was last updated/modified, and the id of the administrator that modified the product. All Dates **must** be stored in GMT format: YYYY-MM-DDTHH:MM:SSZ

Product Discount:
Stores current marked down price for all products that is on sale. It **must** also be support products being offered at a discounted price when buying in bulk.

Technical Design

User Interface

The user interface for this release is to be a **single-page** web application to be built with pure Javascripts or a Javascript framework. The UI will support the following operations.

Admin Operations:
Admin login (Only non-logged in user will see this)
Admin logout (Only logged in user will see this)
Get a listing of all products
Add new product (Only logged in admin users can access this)
Edit a specific product (Only logged in admin users can access this)
Search for product (Only logged in admin users can access this)

UX design of all UI screens.

- This **must** look professional with user-friendly look-and-feel and professional color pallets.
- The UI **must** perform front-end validation and handle back-end responses accordingly--including validation issues on the back-end.
- All UI **must** display user-friendly errors (front-end and back-end) to the user. Alert dialog boxes are not allowed.
- All UI **must** re-display explicit form data to the user, when errors are returned from the back-end during processing.
- The UI **must** support responsive design.
- All UI **must** handle exceptions appropriately. **No page-hanging will ever occur.** This means javascript errors will not cause a page/screen to freeze and stop responding.
- All UI **must** support non-HTML5 compliant browsers.
- HTML forms **must** be submitted by Javascript. Setting the onsubmit attribute in the form tag is not allowed.
- All input button **must** be of type “button”. No “submit” button type is allowed.
- All events **must** be handle by event listeners.
- Minimal in-line CSS. UI look-and-feel must be in separate and external css files.

- When attempting to delete records, the UI **must** confirm the request with the client prior to submitting the request to the service.
- The application **must** fully function in the latest version of Google Chrome, IE, and Firefox.

API Service

The API service will support back-end processing of administrator and production management. The service is to be implemented as a RESTful microservice.

- Every endpoint **must** support JSON request payload
- Every endpoint **must** support JSON response payload
- Every endpoint **must** validate incoming data
- Every endpoint **must** handle errors properly
- Every endpoint **must** handle errors correctly with no exceptions being returned to the client.
- Every endpoint **must** return a standard HTTP response code with an appropriate body payload.
- **Must** support external configurations. (ie, Database connection configuration property values must be read in from external source and not hard coded into the code.

Technology Stack

Front-End

- Javascript (pure or an open-source framework/api)
- HTML (there should only be 1 HTML file).
- CSS

Server-Side (Back-end)

- Php, Java, or NodeJS

Datastore

- RDBMS (MySql or PostgreSQL)

Items to be Submitted

Design Documents

ERD diagram of the proposed data store to support Phase I operations.

Well documented RESTful endpoints. This will include any/all request parameters (require as well as optional parameters). All possible HTTP Response code. The response body format. The request body format, if required.
UX designs of UI to support Phase I of the application.
DDL of all datastore created for Phase.
DML, if any, of test data.
Test Plans
Deployment Instructions

To be Submitted

All design documents
Zipped file of the RESTful service.
Zipped file of the UI implementation code supporting the application.