Inspection Code – Team Loyalty

Core API Code

Product Entity

Java/Spring Boot class for the Product entity.

```
package com.teamloyalty.api.product;

import java.util.Objects;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
class Product {

    private @Id @GeneratedValue(strategy=GenerationType.IDENTITY) Long id;
    @Column(unique=true)
    private String code;
    private String name;
    private String description;
    private Double unitPrice;
    private String imageUrl;
    private String screenshot1;
    private String screenshot2;
    private String level1;
    private String level2;
    private String codeBrand;

    Product() {}

    Product(String code, String name, String description, Double unitPrice, String imageUrl, String
screenshot1, String screenshot2,
            String level1, String level2, String codeBrand) {
        this.code = code;
        this.name = name;
        this.description = description;
        this.unitPrice = unitPrice;
        this.imageUrl = imageUrl;
```

```
        this.screenshot1 = screenshot1;
        this.screenshot2 = screenshot2;
        this.level1 = level1;
        this.level2 = level2;
        this.codeBrand = codeBrand;
    }

    public Long getId() {
        return this.id;
    }

    public String getCode() {
        return this.code;
    }

    public String getName() {
        return this.name;
    }

    public String getDescription() {
        return this.description;
    }

    public Double getUnitPrice() {
        return this.unitPrice;
    }

    public String getImageUrl() {
        return this.imageUrl;
    }

    public String getScreenshot1() {
        return this.screenshot1;
    }

    public String getScreenshot2() {
        return this.screenshot2;
    }

    public String getLevel1() {
        return this.level1;
    }

    public String getLevel2() {
        return this.level2;
```

```
}

public String getCodeBrand() {
    return this.codeBrand;
}

public void setCode(String code) {
    this.code = code;
}

public void setName(String name) {
    this.name = name;
}

public void setDescription(String description) {
    this.description = description;
}

public void setUnitPrice(Double unitPrice) {
    this.unitPrice = unitPrice;
}

public void setImageUrl(String imageUrl) {
    this.imageUrl = imageUrl;
}

public void setScreenshot1(String screenshot1) {
    this.screenshot1 = screenshot1;
}

public void setScreenshot2(String screenshot2) {
    this.screenshot2 = screenshot2;
}

public void setLevel1(String level1) {
    this.level1 = level1;
}

public void setLevel2(String level2) {
    this.level2 = level2;
}

public void setCodeBrand(String codeBrand) {
    this.codeBrand = codeBrand;
}
```

```
    @Override
    public boolean equals(Object o) {
        if (this == o)
            return true;
        if (!(o instanceof Product))
            return false;
        Product product = (Product)o;
        return Objects.equals(this.id, product.id) && Objects.equals(this.code, product.code) &&
Objects.equals(this.name, product.name) &&
                Objects.equals(this.description, product.description) && Objects.equals(this.unitPrice,
product.unitPrice) &&
                Objects.equals(this.imageUrl, product.imageUrl) && Objects.equals(this.screenshot1,
product.screenshot1) &&
                Objects.equals(this.screenshot2, product.screenshot2) && Objects.equals(this.level1,
product.level1) &&
                Objects.equals(this.level2, product.level2) && Objects.equals(this.codeBrand,
product.codeBrand);
    }

    @Override
    public int hashCode() {
        return Objects.hash(this.id, this.code, this.name, this.description, this.unitPrice,
this.imageUrl, this.screenshot1,
                this.screenshot2, this.level1, this.level2, this.codeBrand);
    }

    @Override
    public String toString() {
        return "Product(" + "id=" + this.id + ", code=" + this.code + ", name='" + this.name + "',
description='" + this.description +
                "', unitPrice=" + this.unitPrice + ", imageUrl='" + this.imageUrl + "', screenshot1='" +
this.screenshot1 +
                "', screenshot2='" + this.screenshot2 + "', level1=" + this.level1 + ", level2=" +
this.level2 +
                ", codeBrand=" + this.codeBrand + ")";
    }

}
```

ProductRepository

Java/Spring Boot JPA Repository interface for Product entities.

```java
package com.teamloyalty.api.product;

import java.util.List;
import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;

interface ProductRepository extends JpaRepository<Product, Long> {

    Optional<Product> findByCode(String code);

    List<Product> findByLevel1AndLevel2AndCodeBrand(String level1, String level2, String codeBrand);

    List<Product> findByLevel1AndLevel2(String level1, String level2);

    List<Product> findByLevel1AndCodeBrand(String level1, String codeBrand);

    List<Product> findByLevel2AndCodeBrand(String level2, String codeBrand);

    List<Product> findByLevel1(String level1);

    List<Product> findByLevel2(String level2);

    List<Product> findByCodeBrand(String codeBrand);

}
```

Product Controller

Java/Spring Boot controller class for the Products API

```
package com.teamloyalty.api.product;

import java.util.List;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@CrossOrigin
@RestController
class ProductController {

    private final ProductRepository repository;

    ProductController(ProductRepository repository) {
        this.repository = repository;
    }

    @GetMapping("/products")
    List<Product> all(@RequestParam(name="System", required = false) String level1,
            @RequestParam(name="Genre", required = false) String level2,
            @RequestParam(name="Brand", required = false) String codeBrand) {
        if (level1 != null)
            if (level2 != null)
                if (codeBrand != null)
                    return repository.findByLevel1AndLevel2AndCodeBrand(level1, level2, codeBrand);
                else
                    return repository.findByLevel1AndLevel2(level1, level2);
            else
                if (codeBrand != null)
                    return repository.findByLevel1AndCodeBrand(level1, codeBrand);
                else
                    return repository.findByLevel1(level1);
        else
            if (level2 != null)
                if (codeBrand != null)
                    return repository.findByLevel2AndCodeBrand(level2, codeBrand);
                else
                    return repository.findByLevel2(level2);
            else
                if (codeBrand != null)
```

```
                    return repository.findByCodeBrand(codeBrand);
                else
                    return repository.findAll();

    }

    @GetMapping("/products/{id}")
    Product one(@PathVariable Long id) {
        return repository.findById(id).orElseThrow(() -> new ProductNotFoundException(id));
    }

    // TODO: Revisit this design choice
    @GetMapping("/product")
    Product byCode(@RequestParam String code) {
        return repository.findByCode(code).orElseThrow(() -> new ProductNotFoundException(code));
    }

    @ControllerAdvice
    class ControllerExceptionHandler {
        @ResponseStatus(HttpStatus.NOT_FOUND)
        @ExceptionHandler(ProductNotFoundException.class)
        public ResponseEntity<?> elementNotFound() {
            return ResponseEntity.notFound().build();
        }
    }

}
```

React-Store Browse Products Page

FILENAME: react-store/src/pages/browse.js

DESCRIPTION: This React component displays a set of products retrieved from the API, with dropdown filters dynamically applied by product system, genre and brand.

```
import React from "react";
import "bulma/css/bulma.css";
import { useEffect, useState } from "react";
import GameCard from "../GameCard";

// Browse page

const Browse = () => {
  // set useState to retrieve data from the API
  const [data, setData] = useState([]);
  const [brand, setBrand] = useState([]);
  const [system, setSytems] = useState([]);

  // fetch function to retrieve game cards into the browse
  const fetchData = () => {
    fetch("http://localhost:8080/api/products", {
      mode: "cors",
      method: "GET",
    })
      .then((res) => res.json())
      .then((gameData) => {
        setData(gameData);
      })
      .catch((err) => {
        console.log(err.message);
      });
  };

  // fetch function to retrieve options for dropdown filters
  const fetchBrands = () => {
    fetch("http://localhost:8080/api/brands", {
      mode: "cors",
      method: "GET",
    })
      .then((res) => res.json())
      .then((gameBrands) => {
        setBrand(gameBrands);
```

```
    })
    .catch((err) => {
      console.log(err.message);
    });
};
const fetchSystems = () => {
  fetch("http://localhost:8080/api/productlevels", {
    mode: "cors",
    method: "GET",
  })
    .then((res) => res.json())
    .then((gameSystems) => {
      setSytems(gameSystems);
    })
    .catch((err) => {
      console.log(err.message);
    });
};

// populate state
useEffect(() => {
  fetchData();
}, []);
useEffect(() => {
  fetchBrands();
}, []);
useEffect(() => {
  fetchSystems();
}, []);

const handleChangeSystem = (e) => {
  const { value } = e.currentTarget;
  const genreFilter = document.getElementById("genre-filter").value;
  const brandFilter = document.getElementById("brand-filter").value;
  console.log(genreFilter);
  console.log(brandFilter);
  e.stopPropagation();
  if (value !== "start-here") {
    fetch(`http://localhost:8080/api/products?System=${value.slice(-4)}`, {
      mode: "cors",
      method: "GET",
    })
      .then((res) => res.json())
      .then((filterData) => {
        data.forEach((_data) => {
```

```
          document.getElementById(_data.id).style.display = "none";
        });
        filterData.forEach((item) => {
          data.forEach((_data) => {
            document.getElementById(item.id).style.display = "inline-block";
          });
        });
      });
  } else {
    data.forEach((_data) => {
      document.getElementById(_data.id).style.display = "inline-block";
    });
  }
};
const handleChangeGenre = (e) => {
  const { value } = e.currentTarget;
  e.stopPropagation();
  if (value !== "start-here") {
    fetch(`http://localhost:8080/api/products?Genre=${value}`, {
      mode: "cors",
      method: "GET",
    })
      .then((res) => res.json())
      .then((filterData) => {
        data.forEach((_data) => {
          document.getElementById(_data.id).style.display = "none";
        });
        filterData.forEach((item) => {
          data.forEach((_data) => {
            document.getElementById(item.id).style.display = "inline-block";
          });
        });
      });
  } else {
    data.forEach((_data) => {
      document.getElementById(_data.id).style.display = "inline-block";
    });
  }
};
const handleChangeBrand = (e) => {
  let { value } = e.currentTarget;
  if (value === "Activision") {
    value = "ACTI";
  } else if (value === "Parker Brothers") {
    value = "PBRO";
```

```
  } else if (value === "Imagic") {
    value = "IMAG";
  } else if (value === "M-Network") {
    value = "MNET";
  }
  e.stopPropagation();
  if (value !== "start-here") {
    fetch(`http://localhost:8080/api/products?Brand=${value.toUpperCase()}`, {
      mode: "cors",
      method: "GET",
    })
      .then((res) => res.json())
      .then((filterData) => {
        data.forEach((_data) => {
          document.getElementById(_data.id).style.display = "none";
        });
        filterData.forEach((item) => {
          data.forEach((_data) => {
            document.getElementById(item.id).style.display = "inline-block";
          });
        });
      });
  } else {
    data.forEach((_data) => {
      document.getElementById(_data.id).style.display = "inline-block";
    });
  }
};

return (
  <div>
    <div
      style={{
        display: "flex",
        justifyContent: "center",
        alignItems: "center",
        // height: "90vh",
      }}
    >
      <h1 style={{ fontSize: "42px" }}>Browse Items Here!</h1>
    </div>
    <div style={{ marginLeft: "100px" }}>
      <select
        id="system-filter"
        style={{ fontStyle: "italic", marginRight: "25px" }}
```

```
        onChange={handleChangeSystem}
      >
        {/* SHOW SYSTEM OPTIONS DROPDOWN */}
        <option value="start-here">Filter by System</option>
        {system.map((_system) => {
          return (
            <option
              value={_system.description}
              style={
                _system.hierarchyLevel === 1
                  ? { fontStyle: "normal" }
                  : { display: "none" }
              }
            >
              {_system.description}
            </option>
          );
        })}
      </select>
      <select
        id="genre-filter"
        style={{ fontStyle: "italic", marginRight: "25px" }}
        onChange={handleChangeGenre}
      >
        <option value="start-here">Filter by Genre</option>
        {/* SHOW GENRE OPTIONS DROPDOWN */}
        {system.map((_system) => {
          return (
            <option
              value={_system.levelCode}
              style={
                _system.hierarchyLevel === 2
                  ? { fontStyle: "normal" }
                  : { display: "none" }
              }
            >
              {_system.levelCode}
            </option>
          );
        })}
      </select>
      <select
        id="brand-filter"
        style={{ fontStyle: "italic" }}
        onChange={handleChangeBrand}
```

```
    >
      {/* SHOW BRAND OPTIONS DROPDOWN */}
      <option value="start-here">Filter by Brand</option>
      {brand.map((_brand) => {
        return (
          <option
            value={_brand.description}
            style={{ fontStyle: "normal" }}
          >
            {_brand.description}
          </option>
        );
      })}
    </select>
  </div>
  <div
    style={{
      position: "relative",
      left: "87px",
      top: "0",
      zIndex: -1,
      marginRight: "90px",
    }}
  >
    {data.map((game) => {
      // render all the game cards
      return (
        <div
          id={game.id}
          className="column is-3"
          style={{
            display: "inline-block",
            whiteSpace: "nowrap",
            overflow: "hidden",
            textOverflow: "ellipsis",
          }}
        >
          <GameCard
            title={
              game.name.length > 17
                ? `${game.name.slice(0, 17)}...`
                : game.name
            }
            price={game.unitPrice}
            image={game.imageUrl}
```

```
                description={
                  game.description.length > 25
                    ? `${game.description.slice(0, 25)}...`
                    : game.description
                }
              />
            </div>
          );
        })}
      </div>
    </div>
  );
};

export default Browse;
```