# BilgeAdam Fashion Products Documentation

## Introduction

The BilgeAdam Fashion Products project is a Next.js application designed to showcase fashion products with advanced filtering and sorting capabilities. This documentation provides detailed insights into the thought process, algorithms, and significant decisions made during the development of this project.

## Thought Process

### 1. Initialization and Data Fetching

* Application starts with the initialization of the Next.js project.
* Axios is configured in axios.js to fetch data from the provided JSON URL (fid-task-4-ffront-products.json).
* Used environment variables (NEXT\_PUBLIC\_DATA\_URL and NEXT\_PUBLIC\_CORS\_PROXY) to store sensitive information like API URLs and CORS proxy, separating configuration from the codebase for security and easy configuration changes.
* The FetchAndEnsureData module ensures data fetching, and the fetched data is made available globally through React context.

### 2. ProductList Component

#### Data Processing

* The ProductList component uses the useGetComputedData hook for data processing.
* The hook computes and sorts product data based on applied filters and sorting parameters.
* It efficiently filters products based on price and size criteria and sorts the filtered products.

#### Responsive Design

* The component is designed with responsive layouts for various screen sizes.
* Utilizes custom UI components (ResultCounts, SearchBox, SortBox) for improved user interaction.
* Implements a filtering drawer for mobile devices to enhance user accessibility.

#### Filtering and Sorting

* Provides functionality to filter products by size and sort by price.
* User interactions trigger the corresponding functions for filtering and sorting.

#### Rendering

* Renders the product list based on the processed data.
* Displays the list with a responsive design, adapting to different device widths.

### 3. Statistics Component

#### Data Calculation

* The Statistics component utilizes the useGetStatistics hook for statistical calculations.
* Computes information about brands with the most affordable products, the largest variety of sizes, and the lowest average price for size "32".

#### Rendering

* Displays the calculated statistical information in a structured format.
* Uses custom UI components (StatisticItem) for presentation.

### 4. Custom UI Components

* PropButton, TextButton, CollapseListItem:
  + Designed for flexibility and reusability.
  + Follows Material-UI styling conventions for consistency.
  + Used across the application to maintain a consistent visual appeal.

### 5. Testing

* Comprehensive unit tests using Jest and React Testing Library.
* Covers edge cases, such as empty data scenarios and specific filter combinations.
* Integration tests to ensure different parts of the application work seamlessly.

### 6. Styling and UI/UX

* Applies styles using Material-UI conventions.
* Ensures a consistent and visually appealing design across the application.
* Prioritizes responsive design for a seamless user experience on various devices.

## Project Core Structure

### Routes

* **/productList:**
  + Renders the ProductList component.
  + Displays a list of products with responsive design.
  + Implements filtering and sorting functionalities.
* **/statistics:**
  + Renders the Statistics component.
  + Presents brand-related statistical information based on specific questions.

### Components

#### ProductList

* Displays a list of fashion products.
* Handles filtering and sorting functionalities.
* Utilizes the useGetComputedData hook for computing and sorting data.
* Uses custom UI components (ResultCounts, SearchBox, SortBox) for improved user interaction.
* Implements responsive design, including a drawer for mobile filtering.

#### Statistics

* Presents statistical information about the product data.
* Leverages the useGetStatistics hook to calculate brand-related statistics.
* Displays the information in a structured format using the StatisticItem component.

#### Custom UI Components

* PropButton, TextButton, CollapseListItem:
  + Designed for flexibility and reusability.
  + Follows Material-UI styling conventions for consistency.

### Hooks

#### useGetComputedData

* Responsible for computing and sorting product data based on applied filters and sorting parameters.
* Utilizes useMemo to efficiently compute data only when dependencies (data, filters, sorts) change.
* Filters products based on price and size criteria.
* Sorts the filtered products based on sorting parameters.

#### useGetStatistics

* Calculates statistical information about the product data.
* Determines brands with the most affordable products, the largest variety of sizes, and the lowest average price for a specific size.
* Uses useMemo to optimize calculations and reduce unnecessary recomputations.

### Modules

#### FetchAndEnsureData (DataProvider)

* Ensures data fetching and provides a context for sharing data.
* Fetches product data using the Axios instance configured in axios.js.
* Utilizes React context to share data across components.
* Handles loading and error states during data fetching.

#### Axios Configuration (axios.js)

* Configures an Axios instance for modularity and ease of configuration.
* Handles data fetching from the provided URL.

## Algorithms Used

### 1. useGetComputedData Hook (src/hooks/productList.js)

**Purpose:** Responsible for computing and sorting the product data based on applied filters and sorting parameters.

**Algorithms/Decisions:**

* Utilizes useMemo to efficiently compute data only when dependencies (data, filters, sorts) change.
* Filters products based on price and size criteria.
* Sorts the filtered products based on sorting parameters.

### 2. useGetStatistics Hook (src/hooks/statistics.js)

**Purpose:** Calculates statistical information about the product data.

**Algorithms/Decisions:**

* Determines the brand with the most products under a certain price.
* Identifies the brand offering the largest selection of sizes.
* Finds the brand with the lowest average price for customers wearing size "32".
* Uses useMemo to optimize computations.

### 3. FetchAndEnsureData Module (src/modules/dataProvider.jsx)

**Purpose:** Ensures data fetching and provides a context for sharing data.

**Algorithms/Decisions:**

* Fetches product data using the Axios instance configured in axios.js.
* Utilizes React context to share data across components.
* Handles loading and error states during data fetching.

### 4. ProductList Component (src/components/productList/index.jsx)

**Purpose:** Displays a filtered and sorted list of fashion products.

**Algorithms/Decisions:**

* Uses custom UI components (ResultCounts, SearchBox, SortBox) for improved user interaction.
* Employs useGetComputedData hook for data processing.
* Implements responsive design, including a drawer for mobile filtering.

### 5. Statistics Component (src/components/statistics/index.jsx)

**Purpose:** Presents statistical information about the product data.

**Algorithms/Decisions:**

* Leverages the useGetStatistics hook to calculate brand-related statistics.
* Displays the information in a structured format using the StatisticItem component.

## Significant Decisions

### 1. Component Modularity

* **Objective:** Enhance maintainability and reusability of code.
* **Decision:** Created modular components with clear responsibilities, facilitating easier code updates and future enhancements.

### 2. Context API for State Management

* **Objective:** Manage and share global state efficiently.
* **Decision:** Utilized React context to handle global state, enabling seamless data sharing between components without the need for prop drilling.

### 3. Component Styling Approach

* **Objective:** Achieve a consistent and visually appealing design.
* **Decision:** Adhered to Material-UI styling conventions and created custom styles for a cohesive and professional appearance.

## Key Components

### 1. useGetComputedData Hook (src/hooks/productList.js)

**Purpose:** Responsible for computing and sorting the product data based on applied filters and sorting parameters.

**Algorithms/Decisions:**

* Utilizes useMemo to efficiently compute data only when dependencies (data, filters, sorts) change.
* Filters products based on price and size criteria.
* Sorts the filtered products based on sorting parameters.

### 2. useGetStatistics Hook (src/hooks/statistics.js)

**Purpose:** Calculates statistical information about the product data.

**Algorithms/Decisions:**

* Determines the brand with the most products under a certain price.
* Identifies the brand offering the largest selection of sizes.
* Finds the brand with the lowest average price for customers wearing size "32".
* Uses useMemo to optimize computations.

### 3. FetchAndEnsureData Module (src/modules/dataProvider.jsx)

**Purpose:** Ensures data fetching and provides a context for sharing data.

**Algorithms/Decisions:**

* Fetches product data using the Axios instance configured in axios.js.
* Utilizes React context to share data across components.
* Handles loading and error states during data fetching.

### 4. ProductList Component (src/components/productList/index.jsx)

**Purpose:** Displays a filtered and sorted list of fashion products.

**Algorithms/Decisions:**

* Uses custom UI components (ResultCounts, SearchBox, SortBox) for improved user interaction.
* Employs useGetComputedData hook for data processing.
* Implements responsive design, including a drawer for mobile filtering.

### 5. Statistics Component (src/components/statistics/index.jsx)

**Purpose:** Presents statistical information about the product data.

**Algorithms/Decisions:**

* Leverages the useGetStatistics hook to calculate brand-related statistics.
* Displays the information in a structured format using the StatisticItem component.

### 6. Custom UI Components (src/components/customUI)

**Purpose:** Enhances the visual appeal and interactivity of the application.

**Decisions:**

* PropButton, TextButton, and CollapseListItem are designed for flexibility and reusability.
* Follows Material-UI styling conventions for consistency.

## Testing

### 1. Unit Testing

**Objective:** Ensure code reliability and functionality.

**Decision:** Implemented comprehensive unit tests using Jest and React Testing Library.

**Focus:** Covered edge cases, such as empty data scenarios and specific filter combinations, for robust code.

### 2. Manual Testing

**Objective:** Verify user interface interactions and responsiveness.

**Decision:** Conducted manual testing to confirm a smooth and intuitive user experience.

**Focus:** Checked statistical information accuracy through manual inspection.

The testing phase for the BilgeAdam Fashion Products project was carried out meticulously to ensure both code reliability and a seamless user experience. Comprehensive unit tests were implemented using Jest and React Testing Library, covering various scenarios to guarantee the robustness of the code. Additionally, manual testing was conducted to verify user interface interactions and responsiveness, with a specific focus on validating the accuracy of statistical information through manual inspection.