

The Egyptian E-Learning University  
(EELU)  
**Faculty of Computers & Information Technology**  
**“OnBudget APP”**

**Submitted By:**

No	Name	ID
1	Kerolos Karam Saber	2000487
2	Osama Abdelghani Abdelhameed	2001209
3	Ebram Shereen Alfy	2000553
4	Youssef Gerges Youssef	2000645
5	Omar mohammed Ramadan	2000537
6	Abdelrahman Mohammed Ashraf	2001256
7	Mario Sarhan Abdullah	2000436

**Under Supervision of:**

Dr. Kamal Hamza	Eng. George Sami
Program Director in Faculty of Computers and Information Technology at Egyptian E-Learning University	Assistant Lecturer in Faculty of Computers and Information Technology at Egyptian E Learning University

“This document is presented as partial fulfillment of the requirements for a Bachelor of Computers and Information Technology degree “  
**EELU- Asyut Center 2024**

## **Acknowledgement**

In the beginning, all praise and thanks are due to God for the success and care that He granted us, enabling us to complete our project. We are also indebted to Prof. Dr. Hisham Mohamed Abdel Salam, President of the Egyptian University for E-Learning, for his unwavering support.

Special thanks are extended to Dr. Hisham Hassan, Dean of the Faculty of Computers and Information Technology at the Egyptian University for E-Learning, and Dr. Mohamed Mahmoud, Director of the Study Center in Assiut, for their endless support, guidance, and encouragement throughout the research process.

We would like to express our sincere gratitude to the Computer and Information Technology Program Director and our supervisor, Dr. Kamal Hamza, for his continuous motivation and support during the implementation of this project. His guidance and encouragement from the inception of the idea to its fruition were invaluable.

Furthermore, we extend our heartfelt thanks to Eng. George Sami for his tremendous efforts and exceptional support during our project. His assistance, whether through guidance, training courses, or other means, was instrumental in shaping the problem and providing insights into the solution. Without his support, we would not have been able to complete the project in a timely manner.

# Abstract

In today's digital age, online shopping has become increasingly popular, offering convenience and accessibility to consumers worldwide. However, despite the convenience of online shopping, local stores often face challenges in competing with large e-commerce platforms. During peak shopping seasons or holidays, local stores may struggle to attract customers, leading to decreased foot traffic and sales.

To address this challenge, the proposed solution is to establish OnBudget, an online platform that connects local stores with consumers. The primary objective of OnBudget is to assist local stores in reaching potential customers and increasing sales by providing them with an online presence.

By leveraging technology, OnBudget aims to bridge the gap between local stores and consumers, offering a seamless shopping experience that combines the convenience of online shopping with the unique offerings of local businesses. Through this online platform, consumers can discover nearby stores, browse their products, and make purchases, thereby supporting local businesses and fostering community engagement.

Recognizing the importance of supporting local economies and preserving the vitality of neighborhood businesses, OnBudget aims to serve as a tool for empowering local stores and revitalizing the retail sector.

## Contents

<b>Acknowledgement .....</b>	<b>2</b>
<b>Abstract .....</b>	<b>3</b>
<b>Chapter (1) .....</b>	<b>7</b>
<b>1.1 Introduction:.....</b>	<b>8</b>
<b>Chapter (2) .....</b>	<b>9</b>
<b>2.1 Problem statement:.....</b>	<b>10</b>
<b>2.2 Problem Solution: .....</b>	<b>11</b>
<b>Chapter (3) .....</b>	<b>13</b>
<b>3.1 Project Phases: .....</b>	<b>14</b>
<b>Chapter (4) .....</b>	<b>16</b>
<b>4.1 Participating Technology (Software):.....</b>	<b>17</b>
<b>4.1.1 Figma: .....</b>	<b>17</b>
<b>4.1.2 Visual Studio Code: .....</b>	<b>17</b>
<b>4.1.3 Android Studio:.....</b>	<b>18</b>
<b>4.1.4 Postman: .....</b>	<b>19</b>
<b>4.1.5 Visual Studio: .....</b>	<b>19</b>
<b>4.1.6 Colab Notebook: .....</b>	<b>21</b>
<b>4.1.7 Hugging Face: .....</b>	<b>21</b>
<b>4.1.8 Kaggle: .....</b>	<b>22</b>
<b>4.2 Technologies: .....</b>	<b>23</b>
<b>4.2.1 Flutter: .....</b>	<b>23</b>
<b>4.2.2 Firebase :.....</b>	<b>34</b>
<b>4.2.3 ASP.NET : .....</b>	<b>35</b>
<b>4.2.4 Scikit-learn : .....</b>	<b>37</b>
<b>4.2.5 TensorFlow:.....</b>	<b>39</b>
<b>4.2.6 Stable Diffusion:.....</b>	<b>39</b>
<b>4.2.7 Flask: .....</b>	<b>41</b>
<b>Chapter (5) .....</b>	<b>42</b>
<b>5.1 Mobile Application .....</b>	<b>43</b>

<b>5.1.1 Components:</b> .....	<b>43</b>
<b>5.1.2 OnBoarding screen:</b> .....	<b>64</b>
<b>5.1.3 Registration and Login pages:</b> .....	<b>69</b>
<b>5.1.4 Profile Screen:</b> .....	<b>106</b>
<b>5.1.5 ChatBot .....</b>	<b>115</b>
<b>5.1.6 Generate Image .....</b>	<b>124</b>
<b>5.1.7 Home .....</b>	<b>131</b>
<b>5.1.8 Product Details .....</b>	<b>147</b>
<b>5.2 Web Application .....</b>	<b>162</b>
<b>5.2.1 Register .....</b>	<b>162</b>
<b>5.2.2 Login.....</b>	<b>168</b>
<b>5.2.3 Home .....</b>	<b>172</b>
<b>5.2.4 Cart.....</b>	<b>208</b>
<b>5.2.5 Wishlist.....</b>	<b>216</b>
<b>5.2.6 Profile .....</b>	<b>231</b>
<b>5.2.7 generate images .....</b>	<b>240</b>
<b>5.2.8 ChatBot .....</b>	<b>248</b>
<b>5.3 Back-End .....</b>	<b>258</b>
<b>5.3.1 Introduction.....</b>	<b>258</b>
<b>5.3.2 Solution Structure.....</b>	<b>258</b>
<b>5.3.3 Configuration .....</b>	<b>259</b>
<b>5.3.4 API Documentation .....</b>	<b>266</b>
<b>5.3.5 Repository Pattern.....</b>	<b>270</b>
<b>5.3.6 Service Layer.....</b>	<b>282</b>
<b>5.3.7 DTOs (Data Transfer Objects) .....</b>	<b>302</b>
<b>5.3.8 Controllers .....</b>	<b>309</b>
<b>5.4 AI ,Machine Learning and Deep learning.....</b>	<b>320</b>
<b>5.4.1 Size recommendation: .....</b>	<b>320</b>
<b>5.4.2 Flask endpoint for Size recommendation model: .....</b>	<b>349</b>
<b>5.4.3 Fashion recommendation model: .....</b>	<b>353</b>

<b>5.4.4 Flask endpoint for fashion recommendation model:</b>	<b>380</b>
<b>Chapter 6</b>	<b>383</b>
<b>6.1 Conclusion:</b>	<b>384</b>
<b>6.2 Future work:</b>	<b>385</b>
<b>Chapter 7</b>	<b>386</b>
<b>References:</b>	<b>387</b>

# Chapter (1)

## **1.1 Introduction:**

This graduation project focuses on the application of an e-commerce platform that aims to bridge the gap between online markets and offline stores, streamlining the buying and selling processes. The project utilizes various technologies, including software tools like Figma, Visual Studio Code, Android Studio, Kaggle, Hugging Face, and Jupyter Notebook, as well as frameworks like Flutter, TensorFlow, Scikit-learn, Flask a ASP.NET Core . The implementation includes both a mobile application and a web application, providing convenient access for users. The documentation acknowledges the contributions and support received throughout the project. Chapter 1 introduces the project, while Chapter 2 presents the problem statement and the motivation behind the project. Chapter 3 outlines the different phases of the project, and Chapter 4 discusses the participating technologies in detail. Chapter 5 covers the implementation and layout of both the mobile and web applications. Chapter 6 concludes the project with key findings and highlights areas for future work. Finally, Chapter 7 lists the references used throughout the documentation.

## Chapter (2)

## **2.1 Problem statement:**

In today's shopping landscape, users face numerous challenges that hinder their shopping experience. One significant issue is the struggle to find clothing items that fit within their budget. Without clear guidance, users may overspend or compromise on quality. Additionally, the integration gap between online and offline stores poses a challenge. These channels often operate independently, making it difficult for users to seamlessly transition between them and find the best deals.

Moreover, users receive generic recommendations that do not reflect their individual preferences, resulting in a subpar shopping experience. Furthermore, limited shipping options and unclear policies add frustration and uncertainty to the shopping process. Finally, users often have difficulty finding similar products to those they like, especially when they cannot describe them in words.

## **Solution:**

To address these challenges, the "On Budget" project proposes a comprehensive solution leveraging AI-driven features across web and mobile applications. A unified platform will seamlessly integrate both online and offline stores, allowing users to browse and purchase from any location effortlessly. Additionally, AI-powered recommendation model will analyze user purchase to offer personalized product suggestions tailored to individual preferences. Moreover, the platform will provide transparent shipping policies and recommend the most cost-effective and efficient shipping options using AI algorithms. Lastly, image

recognition technology will enable users to find similar products by uploading images, enhancing product discovery.

## **Expected Outcomes:**

The implementation of these solutions is expected to yield significant improvements in the shopping experience. Users will enjoy a seamless shopping journey with personalized recommendations and budget management features, ultimately leading to higher satisfaction and engagement.

Moreover, the integration of online and offline stores will result in increased sales and user engagement, while transparent shipping policies and efficient shipping options will save users money and reduce delays. Additionally, image recognition technology will make it easier for users to find products they like, further enhancing the overall shopping experience.

Overall, the "On Budget" project aims to revolutionize the shopping experience by empowering users to shop smarter, within their means, and with greater satisfaction, through the integration of AI-driven features and seamless integration of online and offline stores.

## **2.2 Problem Solution:**

To address these challenges, the "On Budget" project proposes a comprehensive solution leveraging AI-driven features across web and mobile applications. A unified platform will seamlessly integrate both online and offline stores, bridging the gap between

local stores and the online market, allowing users to browse and purchase from any location effortlessly. Additionally, an AI-powered recommendation model will analyze user purchases to offer personalized product suggestions tailored to individual preferences, ensuring a more satisfying shopping experience.

Furthermore, the platform will allow users to customize their own t-shirts by entering a prompt, which will then be transformed into a unique t-shirt design using stable diffusion technology. It will also provide size recommendations based on user measurements to ensure a perfect fit. Transparent shipping policies will be presented, with AI algorithms recommending the most cost-effective and efficient shipping options. Users will be able to track their orders and choose from multiple shipping policies to best suit their needs.

Additionally, the platform will feature a chatbot to answer all fashion-related questions, providing immediate assistance and enhancing user experience. Lastly, image recognition technology will enable users to find similar products by uploading images, thereby enhancing product discovery and making the shopping process more intuitive and enjoyable.

## **Chapter (3)**

## **3.1 Project Phases:**

- Phase 1: Study**

During this phase, we aim to transform an abstract idea into a concrete and meaningful goal. It involves conducting a feasibility study and defining the project at a high level. This includes identifying the project's needs, determining project constraints and objectives, creating a projected schedule, and gathering necessary details.

- Phase 2: Analysis**

Analysis can be conducted in a comprehensive or simplified manner, and the information gathered can be valuable for both current and future decision-making. Regularly conducting analysis is beneficial, particularly during times of volatility when information can rapidly change.

- Phase 3: Design**

In this phase, the requirements gathered in the previous stages are documented, and the software architecture necessary for implementing system development is derived.

- Phase 4: Development**

The Development Phase represents a crucial step in constructing the system. The preceding phases lay the groundwork for system development, while the subsequent phases ensure that the system functions as intended.

- **Phase 5: Implementation**

This phase involves testing, inspection, adjustment, correction, and certification of facilities and systems to ensure they perform according to specifications. The implementation phase commences with the notice to proceed for the construction contract and concludes with the final acceptance of the project, unless otherwise specified by grant or regulatory requirements.

- **Phase 6: Validation**

During this stage, the previously designed process is evaluated and qualified to ensure it can consistently and reliably produce the desired level of quality.

- **Phase 7: Follow-up**

The follow-up phase is dedicated to ensuring that all recommendations provided during the initial audit are implemented within the set deadline. It involves dividing the overall process into distinct areas of responsibility and breaking it down further into four sub-phases: status check I, follow-up I, status check II, and follow-up I

## Chapter (4)

## 4.1 Participating Technology (Software):

### 4.1.1 Figma:

Figma is a cloud-based design and prototyping tool used for interface design. It allows designers and developers to collaborate in real-time, making it easier to create, share, and iterate on designs. Figma's features include vector tools, prototyping capabilities, and a wide range of plugins to extend its functionality. In our project, Figma was used to design the UI/UX of both the mobile application and the website, ensuring a consistent look and feel across platforms.

### 4.1.2 Visual Studio Code:

VSCode is a lightweight, yet powerful source-code editor developed by Microsoft. It supports multiple programming languages and frameworks through extensions. Key features include IntelliSense (intelligent code completion), debugging, integrated terminal, and version control support. In our project, VSCode was the primary editor for writing Flutter and Dart code. It provides a seamless and efficient development experience with various extensions specifically designed for Flutter and Dart, such as the Flutter extension, which offers features like hot reload, widget inspection, and code snippets.

### 4.1.3 Android Studio:

Android Studio is the official Integrated Development Environment (IDE) for Android development, based on IntelliJ IDEA. While VSCode was used for writing most of the Flutter code, Android Studio was essential for setting up the SDKs and development environment for Flutter. Android Studio provides tools to:

- Install and manage the Android SDK.
- Configure emulators for testing Flutter applications.
- Access additional tools like the Flutter plugin for Android Studio, which assists in Flutter development by providing features like Flutter Doctor integration, new project templates, and an enhanced UI for developing Flutter apps.

Android Studio was crucial for ensuring that the necessary environments and dependencies were properly configured for Flutter development and for performing tasks such as debugging and profiling the Flutter app on Android devices.

#### **4.1.4 Postman:**

Postman is an API client that simplifies the process of developing, testing, and monitoring APIs. It allows developers to send requests to APIs, inspect responses, and automate testing. Postman was used extensively to test the backend APIs, ensuring they worked correctly with the mobile application and the website.

#### **4.1.5 Visual Studio:**

**Visual Studio** is an integrated development environment (IDE) from Microsoft, widely used for developing applications across various platforms, including web, mobile, desktop, and cloud.

#### **Key Features of Visual Studio**

- Comprehensive Development Tools:** Tools for coding, debugging, testing, and deploying applications.
- IntelliSense:** Advanced code-completion tool for faster coding and reduced errors.
- Debugging:** Powerful debugging capabilities with breakpoints, variable watches, and call stack inspection.
- Integrated Git Support:** Built-in version control using Git for easy code collaboration.
- Extensions and Customization:** Marketplace of extensions for customizable functionality.
- Live Share:** Real-time collaboration feature for sharing codebase and working together seamlessly.
- Azure Integration:** Simplified deployment and management of applications in the cloud.
- Multi-language Support:** Support for multiple programming languages like C#, VB.NET, C++, JavaScript, and Python.

# Setting Up Visual Studio for ASP.NET Core Development

## 1. Download and Install Visual Studio:

- Download from [Visual Studio website](#).
- Select the "ASP.NET and web development" workload during installation.

## 2. Create a New ASP.NET Core Project:

- Open Visual Studio, create a new project, select "ASP.NET Core Web Application," and follow the setup wizard.

## 3. Write and Manage Code:

- Use the editor for writing code with features like syntax highlighting and IntelliSense.

## 4. Debugging and Testing:

- Set breakpoints and start debugging with F5.
- Use the Test Explorer for running unit tests and integration tests.

## 5. Version Control Integration:

- Manage source code with built-in Git tools for committing, pushing, pulling, and merging changes.

## 6. Deploying to Azure:

- Use built-in publishing tools to deploy applications to Azure.

## Benefits of Using Visual Studio

- **Productivity:** Enhances developer productivity with rich tools and features.
- **Collaboration:** Facilitates effective collaboration with Git support and Live Share.
- **Comprehensive Solution:** Provides tools for coding, debugging, testing, and deploying applications.
- **Continuous Improvement:** Regular updates and extensions keep Visual Studio cutting-edge.

#### **4.1.6 Colab Notebook:**

Google Colab is a free cloud-based platform for Python programming, focused on data science and machine learning. It offers an environment similar to Jupyter Notebooks, allowing users to write and execute Python code directly in the browser without any setup. With access to Google's powerful hardware infrastructure, including GPUs and TPUs, users can efficiently train machine learning models and perform computationally intensive tasks. One standout feature is its seamless integration with Google Drive, enabling easy access to files and datasets stored in Google Drive directly from Colab notebooks. Additionally, Colab comes with pre-installed libraries like TensorFlow, PyTorch, and scikit-learn, simplifying the setup process. It also supports real-time collaboration, making it suitable for team projects and educational purposes. Overall, Google Colab provides a convenient platform for data science and machine learning, combining the flexibility of Jupyter Notebooks with Google's cloud infrastructure.

#### **4.1.7 Hugging Face:**

Hugging Face is a prominent platform known for its contributions to machine learning, particularly in the realm of deep learning models. It offers a diverse range of pre-trained models and libraries that facilitate the integration of cutting-edge machine learning capabilities into various applications. Hugging Face's user-friendly interface makes it easy for developers to access and utilize these models, while also providing tools for fine-tuning them to specific tasks. Moreover, the platform has fostered a

vibrant community of developers who actively contribute to the improvement and expansion of available models. This collaborative effort ensures that the latest advancements in machine learning are readily accessible to users across different domains. Overall, Hugging Face serves as a valuable resource for practitioners seeking to leverage state-of-the-art machine learning techniques in their projects.

#### 4.1.8 Kaggle:

Kaggle is a popular platform for data science competitions, datasets, and notebooks. It hosts a wide range of machine learning competitions where data scientists and machine learning practitioners can compete to solve complex problems. Additionally, Kaggle provides a vast repository of datasets covering various topics and domains, allowing users to explore and analyze data for research or personal projects. The platform also offers Kaggle Notebooks, which are an integrated environment for writing and executing code, making it easy to collaborate and share insights with the community. With its active community, rich resources, and collaborative features, Kaggle has become a central hub for data science enthusiasts and professionals alike.

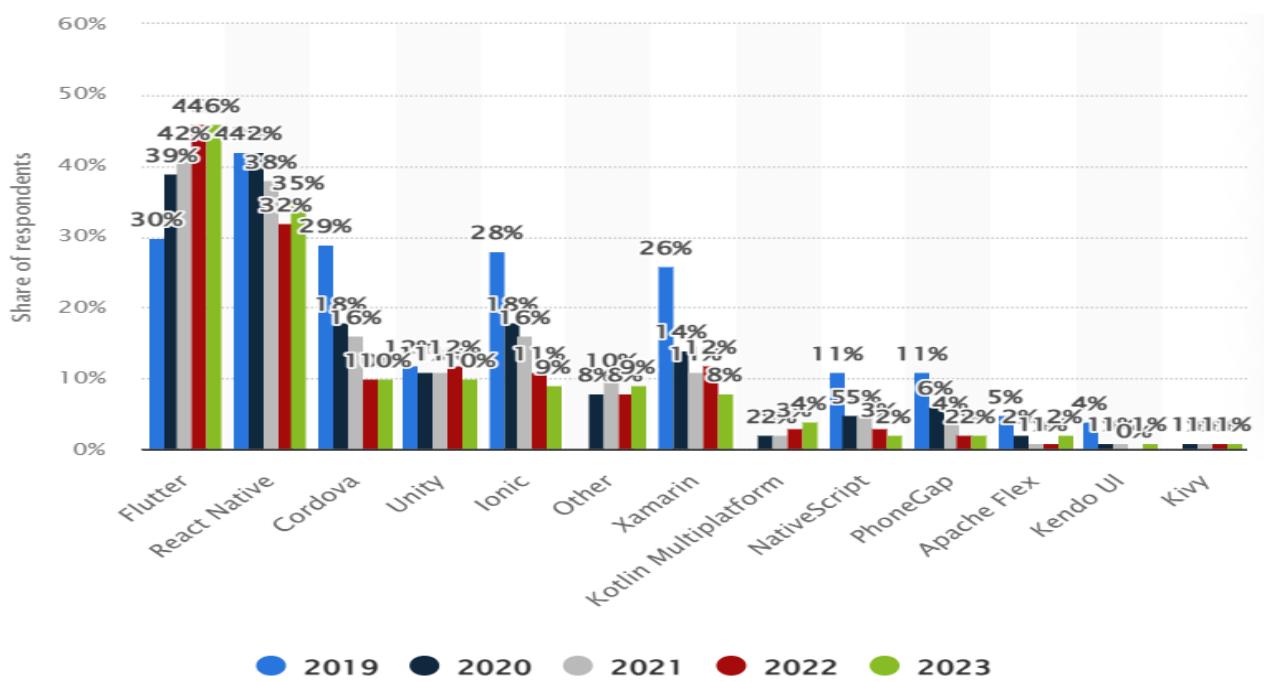
## 4.2 Technologies:

### 4.2.1 Flutter: What is Flutter?

Flutter is an open-source UI software development toolkit created by Google. It is used for building cross-platform applications for mobile, web, and desktop from a single codebase. Flutter utilizes the Dart programming language and offers a rich set of pre-designed widgets that make it easy to create beautiful and responsive user interfaces.

### Flutter Usage Statistics:

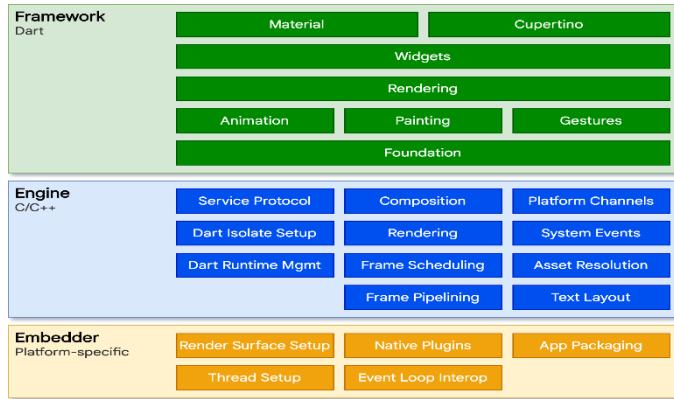
Flutter is the most popular cross-platform mobile framework used by global developers, according to a 2023 developer survey. Based on the survey, 46 percent of software developers used Flutter. On the whole, roughly one third of mobile developers use cross-platform technologies or frameworks; the rest of mobile developers use native tools.



## **Flutter Architecture Explained:**

Flutter's architecture is designed to deliver high performance and flexible UI design. It consists of three main layers:

1. **Framework Layer:** This is the topmost layer, written in Dart, and includes all the essential components like widgets, rendering, gestures, and animation. It is divided into multiple sub-layers:
  - **Widgets:** The building blocks of a Flutter application. They describe the structure, appearance, and behavior of the UI.
  - **Rendering:** Responsible for laying out widgets and painting them on the screen.
  - **Gestures:** Manages user input and touch events.
  - **Animation:** Handles the animations and transitions within the app.
2. **Engine Layer:** This middle layer is written in C++ and provides low-level rendering support using the Skia graphics library. It handles tasks such as:
  - **Dart Runtime:** Executes the Dart code.
  - **Text Layout:** Manages text rendering and layout.
  - **Accessibility:** Provides support for accessibility features.
3. **Embedder Layer:** This bottom layer interacts with the underlying operating system. It includes platform-specific code for iOS, Android, Windows, macOS, Linux, and the web. It handles tasks such as:
  - **Platform Channels:** Facilitates communication between Dart code and platform-specific code.
  - **Event Loop:** Manages the event loop for the application.
  - **Rendering Surface:** Provides a surface to render the UI.



## Advantages of Flutter for Developers:

**Hot Reload:** Allows developers to see the changes they make in the code immediately reflected in the app without restarting it. This speeds up the development process.

**Single Codebase:** Enables writing a single codebase for multiple platforms, reducing development time and effort.

**Rich Set of Widgets:** Flutter provides a wide range of customizable widgets to create expressive and flexible UIs.

**Fast Development:** The combination of hot reload, a rich set of widgets, and a robust framework allows for quick prototyping and faster development cycles.

**Strong Community:** Flutter has an active community that contributes to a rich ecosystem of packages and plugins, providing solutions to common problems and extending the framework's capabilities.

## Reasons for Flutter's Success:

**Performance:** Flutter apps are compiled directly to native code, ensuring high performance and a smooth user experience.

**Flexibility:** The widget-based architecture allows for extensive customization and flexibility in UI design.

**Community and Ecosystem:** A strong and active community contributes to a rich ecosystem of packages, plugins, and tools, making it easier to develop complex applications.

**Support from Google:** Being backed by Google, Flutter receives continuous updates, improvements, and long-term support, ensuring its relevance and reliability.

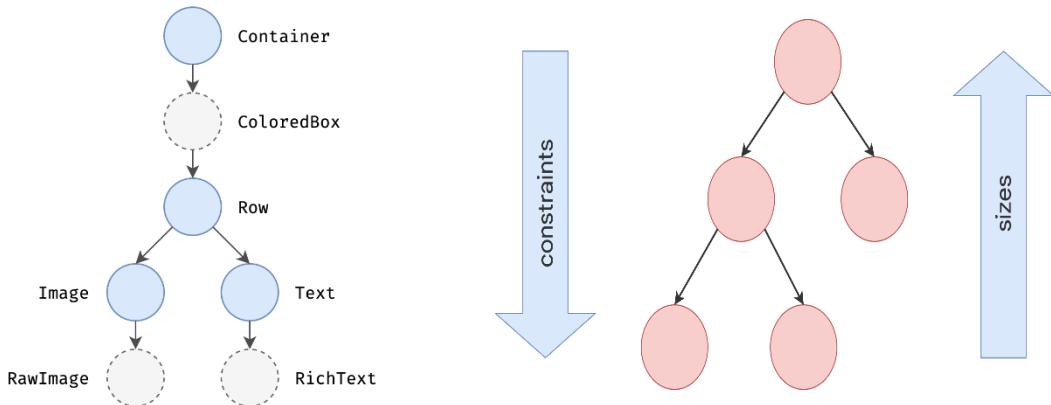
## **Flutter's Highly Customizable and Flexible:**

Flutter's architecture is based on a modern, reactive framework. Its widget-based approach allows for building complex UIs from a collection of simple widgets, which can be easily customized or extended. This flexibility enables developers to create unique and visually appealing designs tailored to specific project requirements.

## **Widget hierarchy in Flutter:**

Flutter's widget hierarchy is a key concept that enables developers to build complex and responsive user interfaces. Widgets are the basic building blocks of a Flutter app's UI and can be nested within one another to create complex layouts. The hierarchy can be broken down into three main types of widgets:

1. **Stateless Widgets:** These widgets do not store any state and are immutable. They are typically used for static content that does not change over time. Examples include Text, Container, and Image.
2. **Stateful Widgets:** These widgets maintain state that can change during the lifetime of the widget. They are used for dynamic content that needs to be updated based on user interaction or other events. Examples include Checkbox, Slider, and TextField. A stateful widget consists of two classes:
  - **StatefulWidget:** The widget itself, which is immutable.
  - **State:** A mutable class that holds the widget's state and logic.
3. **Inherited Widgets:** These widgets allow state to be efficiently passed down the widget tree. They are used to share data across multiple widgets without the need to explicitly pass the data through constructors. Examples include InheritedWidget and Provider (from the Provider package).



## Cross-Platform vs. Native:

### 1. Cross-Platform:

#### Pros:

- Optimal performance and access to all native APIs.
- Better integration with platform-specific functionalities.
- Smaller app size as it only includes platform-specific code.

#### Cons:

- Potential performance overhead compared to native apps.
- Limited access to some platform-specific features and APIs.
- Larger app size due to inclusion of the cross-platform framework.

### 2. Native:

#### Pros:

- Optimal performance and access to all native APIs.
- Better integration with platform-specific functionalities.
- Smaller app size as it only includes platform-specific code.

#### Cons:

- Separate codebases for each platform, increasing development time and costs.
- Higher maintenance effort due to the need to sync features and updates across platforms.

## The Advantages of Flutter over Other Frameworks:

### 1. Flutter vs. React:

- **Performance:** Flutter generally offers better performance because it compiles directly to native code, while React Native uses a JavaScript bridge.
- **UI Consistency:** Flutter's widget-based approach provides a consistent UI across platforms, whereas React Native relies on native components.
- **Development Speed:** Flutter's hot reload feature enhances development speed, allowing for faster iteration.

### 2. Flutter vs. Xamarin:

- **Performance:** Flutter's architecture generally results in better performance compared to Xamarin, which relies on a mix of native and interpreted code.
- **UI Design:** Flutter offers a richer set of customizable widgets, providing more flexibility in UI design.
- **Community:** Flutter has a larger and more active community compared to Xamarin.

### 3. Flutter vs. Ionic:

- **Performance:** Flutter apps perform better as they compile to native code, while Ionic apps run inside a WebView, which can result in slower performance.
- **UI:** Flutter provides a more native-like look and feel, whereas Ionic relies on web technologies.

### 4. Flutter vs. Kotlin Multiplatform:

- **UI Development:** Flutter is more focused on UI development, providing a comprehensive toolkit for building rich UIs.
- **Community and Resources:** Flutter has a larger community and more resources available for developers.

## **What is Responsive Design in flutter?**

Responsive design in Flutter ensures that the app's UI adapts to various screen sizes, orientations, and resolutions. This involves dynamically adjusting layouts, fonts, and other UI elements to provide an optimal user experience across different devices, from small smartphones to large tablets and desktop screens.

## **Ways to make our App Responsive:**

- **Flexible Widgets:** Using widgets like Flexible and Expanded to create adaptable layouts that change based on the available space.
- **MediaQuery:** Accessing device screen properties to adjust layout and styling dynamically. MediaQuery provides information about the screen size, orientation, and pixel density, allowing the app to make responsive adjustments.
- **AspectRatio:** Ensuring UI elements maintain a consistent aspect ratio across different screen sizes. The AspectRatio widget helps in keeping the width-to-height ratio of widgets consistent.
- **Responsive Breakpoints:** Defining different UI layouts for various screen sizes (e.g., mobile, tablet, desktop) and using conditional logic to switch between these layouts based on the screen dimensions.
- **flutter\_screenutil Package:** Using the “flutter\_screenutil” package to simplify responsive design. This package allows developers to define width and height dimensions with the .w and .h extensions, making it easier to scale UI elements based on screen size.

## **Why We Choose the “flutter\_screenutil” Package?**

The “flutter\_screenutil” package was chosen for its ease of use and ability to simplify the process of making the app responsive. This package allows developers to easily scale dimensions and fonts based on the screen size and density, ensuring a consistent look and feel across different devices without manually calculating sizes and adjustments.

## Example of usage:



## Advantages of “flutter\_screenutil”:

- **Ease of Use:** Provides an intuitive API for defining responsive dimensions, reducing the complexity of responsive design.
- **Consistency:** Ensures that UI elements scale consistently across different screen sizes and resolutions.
- **Efficiency:** Minimizes the amount of code needed for responsive design, enhancing development speed and reducing the likelihood of errors.
- **Flexibility:** Offers flexible scaling options for both width and height, giving developers granular control over UI layout.

## What is State Management?

State management refers to the practice of managing the state of an application, which includes the data and UI state. In Flutter, state management ensures that the UI accurately reflects the current state of the app and updates appropriately in response to user interactions or data changes.

## Why Use Cubit as State Management?

Cubit, part of the Bloc library, is a lightweight state management solution that provides a simple and predictable way to manage state transitions. It

is particularly suitable for smaller and medium-sized applications where simplicity and ease of use are essential. Key reasons for using Cubit include:

- **Simplicity:** Offers a straightforward API that is easy to understand and use, reducing the complexity of state management.
- **Predictability:** Provides a clear pattern for state transitions, making it easier to debug and maintain the application.
- **Integration with Bloc:** Allows for easy migration to Bloc if more complex state management is needed in the future.

## Cubit vs. other States Management:

### 1. Cubit

- **Pros:** Simple API, predictable state transitions, part of the Bloc library, minimal boilerplate.
- **Cons:** May require more code for larger applications compared to some other state management solutions.

### 2. Provider

- **Pros:** Easy to use, integrates well with Flutter's widget tree, good for simple state management.
- **Cons:** Can become complex for larger applications, less predictable state transitions compared to Cubit.

### 3. GetX

- **Pros:** Highly efficient, minimal boilerplate, reactive programming model, built-in dependency injection.
- **Cons:** Can lead to tightly coupled code, steep learning curve for complex applications.

### 4. RiverPod

- **Pros:** Improved version of Provider, robust and scalable, supports both synchronous and asynchronous state management.
- **Cons:** Slightly steeper learning curve, more boilerplate compared to Provider.

## Why Choosing Cubit?

Cubit was chosen for its simplicity and ease of use, making it an ideal choice for our project's state management needs. Its integration with the Bloc library ensures that we can easily scale to more complex state management if necessary. Cubit provides a clear and predictable way to manage state transitions, enhancing the maintainability and debuggability of the code.

## What is Clean Architecture?

Clean Architecture is a design pattern that promotes separation of concerns and organizes code into layers, each with a specific responsibility. This approach helps to create a codebase that is more modular, testable, and scalable. The primary goal of Clean Architecture is to ensure that the business logic is independent of frameworks, UI, databases, and external services, making the code easier to maintain and extend.

## What is Clean Architecture We Used in project?

Our project follows a three-layer architecture, aligning with the principles of Clean Architecture and using the Model-View-ViewModel (MVVM) pattern:

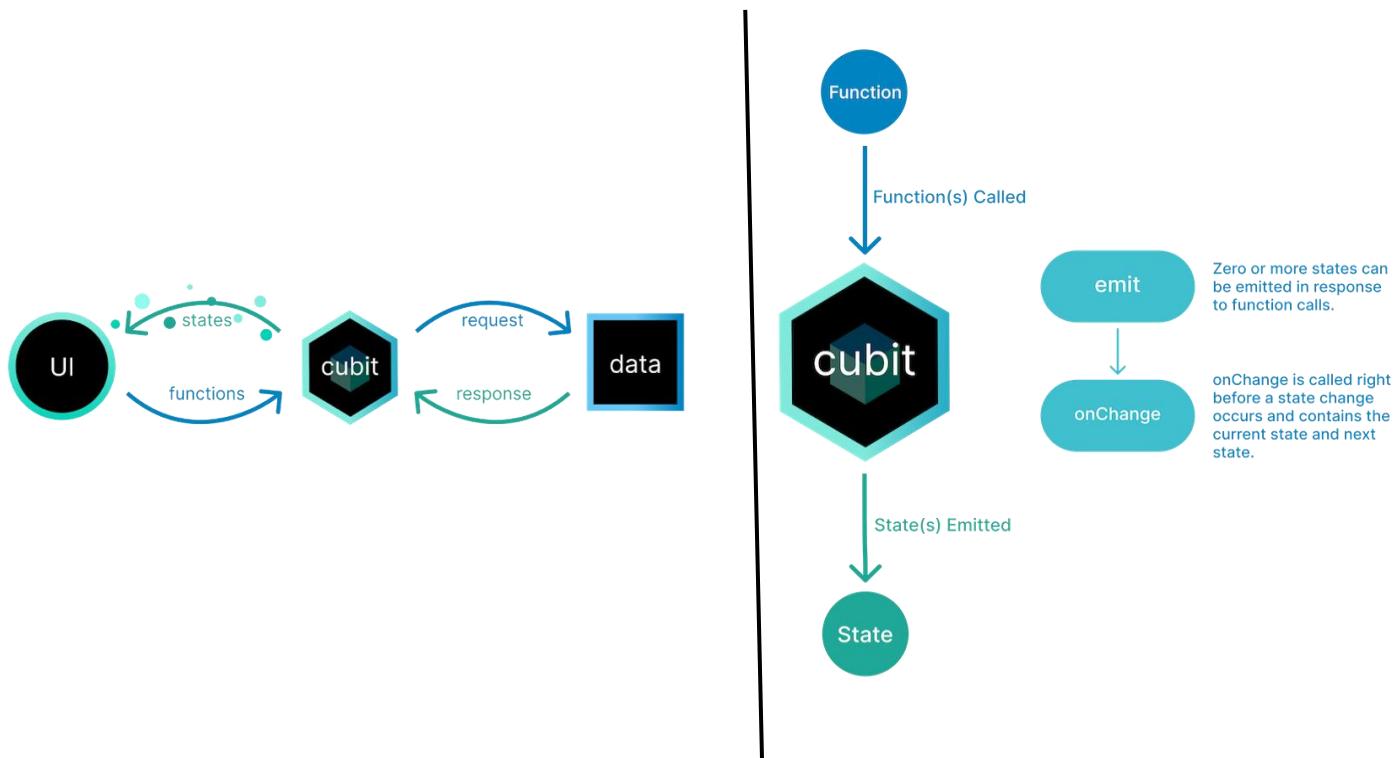
- 1. Data Layer:** Responsible for data management, including three main folders:
  - **Web Service:** Contains code for interacting with external APIs and data sources.
  - **Repository:** Acts as an intermediary between the web service and the rest of the application, providing a clean API for data access.
  - **Models:** Defines the data structures used throughout the application.
- 2. Logic Layer:** Manages the application's business logic and state, consisting of:

- **Cubit File:** Contains business logic for managing state transitions.
- **State File:** Defines the different states of the application, such as success, waiting, and failure states.

**3. Presentation Layer:** Handles the UI and user interactions, organized into:

- **Views:** Contains the screens and pages of the application.
- **Widgets:** Includes reusable UI components.

Each module in the project follows this structure, ensuring consistency and separation of concerns across the application.



## Reason For Using Clean architecture:

Using Clean Architecture provides several benefits:

- **Modularity:** Separation of concerns makes it easier to manage and understand different parts of the codebase.

- **Testability:** Decoupled business logic and data management make it easier to write unit tests and integration tests.
- **Scalability:** A modular architecture makes it easier to extend and modify the application as requirements evolve.
- **Maintainability:** Clear boundaries between layers make it easier to maintain and refactor the code.

Without Clean Architecture, the project would likely face issues such as increased complexity, difficulty in testing, and challenges in scaling the application. A tightly coupled codebase would be harder to maintain and more prone to bugs and regressions.

#### 4.2.2 Firebase :

##### **What is Firebase?**

Firebase is a comprehensive platform developed by Google for building and managing mobile and web applications. It offers a suite of tools and services, including real-time databases, authentication, cloud storage, and hosting, making it a versatile solution for developers. Firebase simplifies the development process by providing a unified backend, allowing developers to focus more on creating and improving their applications rather than managing infrastructure.

##### **Advantages of Firebase:**

One of Firebase's significant advantages is its seamless integration with machine learning (ML) models. By leveraging Firebase, developers can deploy ML models on the cloud and easily integrate them into their applications, enabling real-time predictions and enhanced user experiences without extensive backend infrastructure. Additionally,

Firebase's robust analytics and A/B testing tools allow for continuous monitoring and optimization of ML models, ensuring applications remain responsive and efficient. Overall, Firebase simplifies the deployment, management, and scaling of ML models, accelerating development and reducing operational complexity.

#### 4.2.3 ASP.NET :

##### What is ASP.NET?

ASP.NET is an open-source, server-side web-application framework designed for web development to produce dynamic web pages, web applications, and web services. Developed by Microsoft, it allows developers to build robust, scalable, and secure applications using the .NET platform.

##### Key Features of ASP.NET

- 1. Cross-Platform:** ASP.NET Core runs on Windows, Linux, and macOS, providing developers the flexibility to deploy applications across different operating systems.
- 2. High Performance:** ASP.NET Core is known for its high performance and low memory consumption, making it one of the fastest web frameworks available.
- 3. Unified Framework:** ASP.NET Core combines the features of MVC (Model-View-Controller) and Web API into a single framework, offering a unified programming model for building web applications and services.
- 4. Dependency Injection:** Built-in support for dependency injection (DI) allows for better modularity, testability, and maintainability of code.

5. **Modular and Lightweight:** The framework is designed to be modular, enabling developers to include only the necessary components, thus reducing the application's footprint.
6. **Asynchronous Programming:** ASP.NET Core supports asynchronous programming patterns, improving the scalability of web applications by efficiently handling multiple simultaneous requests.
7. **Security:** ASP.NET Core provides robust security features, including built-in support for authentication, authorization, and data protection.
8. **Cloud Integration:** Seamlessly integrates with cloud platforms like Microsoft Azure, making it easier to deploy, manage, and scale applications.

## Components of ASP.NET

1. **ASP.NET Core MVC:** A framework for building web applications using the Model-View-Controller design pattern. It promotes a clean separation of concerns and enhances testability and maintainability.
2. **Razor Pages:** A page-centric programming model that simplifies the development of web applications by providing a single-file structure that contains both the page's view and logic.
3. **Blazor:** A framework for building interactive web UIs using C#. Blazor WebAssembly allows running C# code directly in the browser, while Blazor Server runs on the server.
4. **ASP.NET Web API:** A framework for building HTTP services that can be consumed by various clients, including browsers, mobile devices, and desktop applications.
5. **SignalR:** A library for adding real-time web functionality to applications, allowing server-side code to push updates to clients instantly.

## **Use Cases of ASP.NET**

- **Web Applications:** Developing dynamic, data-driven websites with rich user interfaces.
- **Web APIs:** Creating RESTful services that can be consumed by various clients, including mobile apps, single-page applications (SPAs), and other web services.
- **Real-Time Applications:** Building applications that require real-time communication, such as chat applications, live notifications, and online gaming.
- **Enterprise Applications:** Developing large-scale enterprise applications with complex business logic, security requirements, and integration needs.

## **Benefits of Using ASP.NET**

- **Performance:** Optimized for speed and efficiency, providing fast response times and high throughput.
- **Versatility:** Suitable for a wide range of applications, from simple websites to complex enterprise solutions.
- **Scalability:** Can handle large-scale applications and high-traffic scenarios with ease.
- **Security:** Built-in features for protecting applications against common threats, such as cross-site scripting (XSS) and cross-site request forgery (CSRF).
- **Community and Support:** Backed by Microsoft and a vibrant community, ensuring continuous improvements, updates, and a wealth of resources.

### **4.2.4 Scikit-learn :**

#### **What is scikit-learn?**

Scikit-learn, also known as sklearn, is a widely used open-source machine learning library for Python. It offers a simple and consistent interface for various machine learning algorithms, making it accessible to both beginners and experienced practitioners. With comprehensive documentation and numerous practical examples, scikit-learn facilitates learning and experimentation in the field of machine learning. Additionally, it provides a wide range of functionalities, including supervised and unsupervised learning algorithms, model selection, evaluation, and preprocessing tools. Its optimized implementations of algorithms and data structures ensure efficient performance, even for large datasets. Seamless integration with other Python libraries such as NumPy, SciPy, and Matplotlib further enhances its capabilities, making scikit-learn a go-to choice for many data scientists and machine learning engineers.

### **Advantages of scikit-learn:**

The advantages of scikit-learn lie in its ease of use and comprehensive functionality. Its simple and consistent interface allows users to quickly build and deploy machine learning models without the need for extensive coding or technical expertise. Moreover, scikit-learn offers a wide range of algorithms and tools for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, and more. Its performance and scalability are also notable, as it is built upon optimized implementations of algorithms and data structures. Furthermore, scikit-learn seamlessly integrates with other Python libraries such as pandas, enabling efficient data manipulation and analysis. Overall, scikit-learn provides a powerful and versatile platform for machine learning development, making it an indispensable tool for data scientists and machine learning practitioners.

## 4.2.5 TensorFlow:

### What is TensorFlow?

TensorFlow, developed by Google, is a widely-used open-source machine learning framework known for its flexibility and scalability. It offers a comprehensive ecosystem of tools and libraries for building and deploying machine learning models, particularly deep learning models. From simple linear regressions to complex neural networks for tasks like image recognition and natural language processing, TensorFlow provides the necessary resources to tackle various machine learning challenges. Its computational graph abstraction allows for efficient execution across CPUs, GPUs, and distributed computing environments, making it suitable for training models on large datasets and deploying them in production settings.

### Advantages of TensorFlow:

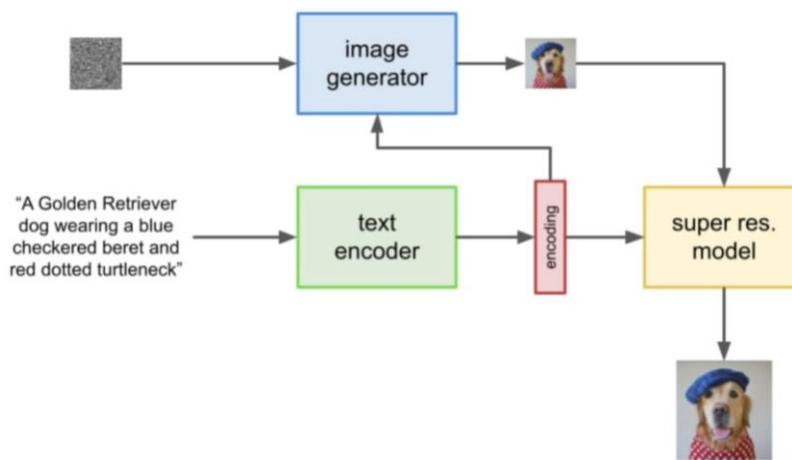
One of TensorFlow's key advantages lies in its extensive community support and adoption in both academia and industry. This widespread use has led to a wealth of resources, including tutorials, documentation, and pre-trained models, which facilitate learning and development. Additionally, TensorFlow's scalability and efficiency enable developers to tackle complex computational tasks, driving innovation in machine learning and enabling the creation of cutting-edge AI applications.

## 4.2.6 Stable Diffusion:

Stable Diffusion is a generative model that leverages the principles of diffusion models to create data, particularly images, that are similar to a set of training data. It operates by iteratively adding Gaussian noise to an

image and then learning to reverse this process to recover the original image. Unlike traditional diffusion models, which may be computationally intensive and slow, Stable Diffusion incorporates several optimizations to improve stability and efficiency, making it more practical for real-world applications.

## Stable Diffusion



### Advantages of Stable Diffusion:

Stable Diffusion offers significant advantages, making it a powerful tool in generative modeling. It provides enhanced stability, overcoming the instability issues of traditional diffusion models to ensure reliable, consistent high-quality image generation. Its computational efficiency allows for faster image generation, suitable for real-time applications. The model produces high-fidelity images that closely resemble the training data, valuable for tasks like artistic creation and medical imaging. Additionally, Stable Diffusion is scalable to large datasets and high-resolution images, making it versatile across various tasks. Its robustness to noise, achieved through training on the process of adding and removing noise, ensures good performance even with varying data quality. These

advantages make Stable Diffusion an efficient, reliable, and high-quality solution for image synthesis and other generative tasks.

#### 4.2.7 Flask:

##### What is Flask?

Flask is an excellent choice for serving machine learning models through a web interface. By using Flask as an endpoint for an ML model, developers can create RESTful APIs that handle requests and responses seamlessly. This setup allows the ML model to be accessed remotely, enabling real-time predictions and data processing over the web. Flask's lightweight nature ensures that the application remains fast and responsive, even when handling complex computations. Additionally, Flask provides robust support for handling various HTTP methods, such as GET and POST, which are essential for sending input data to the model and retrieving the output.

##### Advantages of Flask:

The framework's simplicity and flexibility make it easy to integrate with other tools and services, such as databases and authentication mechanisms, ensuring a secure and scalable deployment of the ML model. Furthermore, Flask's extensive documentation and active community offer valuable resources for building, deploying, and maintaining ML endpoints, making it a preferred choice for many developers in the field. Flask's lightweight nature ensures that the application remains fast and responsive, even when handling complex computations.

## Chapter (5)

# Implementation & Layout

## 5.1 Mobile Application

### 5.1.1 Components:

Bottom Navbar: A navigation bar placed at the bottom of the screen for easy access to app sections.

```
// ignore_for_file: public_member_api_docs, sort_constructors_first
import
'package:animated_notch_bottom_bar:animated_notch_bottom_bar)animated_notch_bottom_bar.dart';
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:flutter_svg/svg.dart';
import
'package:on_budget/src/modules/customer_modules/chatbot/presentation/views/chatbot.dart';
import
'package:on_budget/src/modules/customer_modules/generate_image/presentation/views/generate_t-shirt.dart';
import
'package:on_budget/src/modules/customer_modules/home/presentation/views/home.dart';
import
'package:on_budget/src/modules/customer_modules/home/presentation/views/search.dart';
import 'package:on_budget/src/modules/profile/presentation/views/profile.dart';
import
'package:on_budget/src/utils/components/bottom_bar/scroll_to_hide_widget.dart';
import 'package:on_budget/src/utils/helper/constants/colors.dart';
import 'package:on_budget/src/utils/helper/constants/images.dart';

// ignore: must_be_immutable
class CustomBottomNavBar extends StatelessWidget {
  int currentIndex;
  ScrollController controller;
  final int lastIndex;
  CustomBottomNavBar({
    Key? key,
    required this.currentIndex,
    required this.controller,
```

```

        required this.lastIndex,
    }) : super(key: key);

    void _handleTap(BuildContext context, int index) {
        switch (index) {
            case 0:
                currentIndex = 0;
                if (currentIndex != lastIndex) {
                    Navigator.pushNamed(context, Home.id);
                }
                break;
            case 1:
                currentIndex = 1;
                if (currentIndex != lastIndex) {
                    Navigator.pushNamed(context, SearchScreen.id);
                }
                break;
            case 2:
                currentIndex = 2;
                if (currentIndex != lastIndex) {
                    Navigator.pushNamed(context, ChatBotView.id);
                }
            case 3:
                currentIndex = 3;
                if (currentIndex != lastIndex) {
                    Navigator.pushNamed(context, GenerateTshirt.id);
                }
                break;
            case 4:
                currentIndex = 4;
                if (currentIndex != lastIndex) {
                    Navigator.pushNamed(context, ProfileScreen.id);
                }
                break;
        }
    }

    @override
    Widget build(BuildContext context) {
        return Align(
            alignment: Alignment.bottomCenter,
            child: ScrollToHideWidget(
                controller: controller,
                duration: const Duration(milliseconds: 700),

```

```

        child: AnimatedNotchBottomBar(
            notchBottomBarController:
                NotchBottomBarController(index: currentIndex),
            color: Colors.white,
            showLabel: true,
            elevation: 1,
            bottomBarItems: [
                const BottomBarItem(
                    inActiveItem: Icon(Icons.home_filled, color: Colors.blueGrey),
                    activeItem: Icon(Icons.home_filled, color: kPrimaryColor),
                    itemLabel: 'Home',
                ),
                const BottomBarItem(
                    inActiveItem: Icon(Icons.search, color: Colors.blueGrey),
                    activeItem: Icon(Icons.search, color: kPrimaryColor),
                    itemLabel: 'Search',
                ),
                BottomBarItem(
                    inActiveItem: SvgPicture.asset(
                        kGeminiSparkleInActive,
                    ),
                    activeItem: SvgPicture.asset(kGeminiSparkleActive),
                    itemLabel: 'Tips',
                ),
                BottomBarItem(
                    inActiveItem: SvgPicture.asset(kGenerateImagesInActive),
                    activeItem: SvgPicture.asset(kGenerateImagesActive),
                    itemLabel: 'Inspiration',
                ),
                const BottomBarItem(
                    inActiveItem: Icon(Icons.person, color: Colors.blueGrey),
                    activeItem: Icon(Icons.person, color: kPrimaryColor),
                    itemLabel: 'Profile',
                ),
            ],
            onTap: (index) {
                _handleTap(context, index);
            },
            kBottomRadius: 28.0.r,
            kIconSize: 25.r,
        ),
    ),
);
}
}

```

**Scroll to Hide Bottom Navbar:** A widget that automatically hides the bottom navigation bar when the user scrolls down and shows it again when scrolling stops.

```
import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';

class ScrollToHideWidget extends StatefulWidget {
  const ScrollToHideWidget({
    Key? key,
    required this.child,
    required this.controller,
    this.duration = const Duration(milliseconds: 600),
  }) : super(key: key);
  final Widget child;
  final ScrollController controller;
  final Duration duration;

  @override
  State<ScrollToHideWidget> createState() => _ScrollToHideWidgetState();
}

class _ScrollToHideWidgetState extends State<ScrollToHideWidget> {
  @override
  void initState() {
    // TODO: implement initState
    super.initState();
    widget.controller.addListener(listen);
  }

  @override
  void dispose() {
    // TODO: implement dispose
    super.dispose();
    widget.controller.removeListener(listen);
  }

  void listen() {
    final direction = widget.controller.position.userScrollDirection;
    if (direction == ScrollDirection.forward) {
      show();
    } else if (direction == ScrollDirection.reverse) {
      hide();
    }
  }
}
```

```
void show() {
    if (!isVisible) {
        setState(() {
            isVisible = true;
        });
    }
}

void hide() {
    if (isVisible) {
        setState(() {
            isVisible = false;
        });
    }
}

bool isVisible = true;
@Override
Widget build(BuildContext context) => AnimatedContainer(
    duration: widget.duration,
    height: isVisible ? 100 : 0,
    child: Wrap(
        children: [widget.child],
    ),
);
}
```

**Profile Section:** A customizable widget for the profile screen, featuring a row with an icon, text, and arrow for easy navigation.

```
import 'package:flutter/material.dart';

import '../../../../../modules/profile/presentation/views/adresses/show_addresses.dart';
import '../../../../../modules/profile/presentation/views/copoun/empty_copoun.dart';
import '../../../../../modules/profile/presentation/views/notifications/empty_notification.dart';
import '../../../../../modules/profile/presentation/views/orders/empty_order.dart';
import '../../../../../modules/profile/presentation/views/wishlist/empty_wishlist.dart';
import '../../../../../modules/profile/presentation/widgets/profile_sections.dart';
import '../../../../../modules/register/login/presentation/views/login.dart';
import '../show_dialog.dart';

class GetProfileSections {
    BuildContext context;
    GetProfileSections({required this.context});

    List<ProfileSections> buildProfileSections() {
        return [
            ProfileSections(
                icon: Icons.notifications_none,
                text: 'Notifications',
                onTap: () => Navigator.pushNamed(context, EmptyNotification.id)),
            ProfileSections(
                icon: Icons.discount,
                text: 'Copouns',
                onTap: () => Navigator.pushNamed(context, EmptyCoupon.id)),
            ProfileSections(
                icon: Icons.receipt,
                text: 'My orders',
                onTap: () => Navigator.pushNamed(context, EmptyOrders.id)),
            ProfileSections(
                icon: Icons.near_me,
                text: 'Addresses',
                onTap: () => Navigator.pushNamed(context, ShowAddresses.id)),
            ProfileSections(icon: Icons.payment, text: 'Payment', onTap: () {}),
            ProfileSections(
                icon: Icons.favorite_border,
                text: 'Wishlist',
                onTap: () => Navigator.pushNamed(context, EmptyWishlist.id)),
        ];
    }
}
```

```
ProfileSections(
    icon: Icons.login_outlined,
    text: 'Logout',
    onTap: () {
        showDialogWidget(
            context,
            firstTitle: 'Are you sure',
            secondTitle: 'you want to logout?',
            firstTap: () => Navigator.pop(context),
            secondTap: () => Navigator.pushNamed(context, Login.id),
            firstButtonText: 'Cancel',
            secondButtonText: 'Sure',
        );
    },
),
ProfileSections(
    icon: Icons.person_remove_alt_1_outlined,
    text: 'Delete Account',
    onTap: () {
        showDialogWidget(context,
            firstTitle: 'Are you sure',
            secondTitle: 'you want to delete your account?',
            firstTap: () => Navigator.pop(context),
            secondTap: () => Navigator.pushNamed(context, Login.id),
            firstButtonText: 'Cancel',
            secondButtonText: 'Delete',
            firstTitleSize: 20,
            secondTitleSize: 20);
    },
),
];
}
}
```

**Background Widget:** Allows customization of background shapes for app bars and body sections across all screens.

```
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

import '../helper/constants/colors.dart';

class Background extends StatelessWidget {
    const Background(
        {super.key,
        required this.child,
        this.horizontalPadding,
        this.verticalPadding});

    final Widget child;
    final double? horizontalPadding;
    final double? verticalPadding;

    @override
    Widget build(BuildContext context) {
        return Container(
            color: kPrimaryColor,
            width: double.infinity,
            height: double.infinity,
            child: Container(
                decoration: BoxDecoration(
                    borderRadius: const BorderRadius.only(
                        topLeft: Radius.circular(20), topRight: Radius.circular(20))
                    .w,
                    color: kThirdColor,
                ),
                child: Padding(
                    padding: EdgeInsets.symmetric(
                        horizontal: horizontalPadding ?? 20,
                        vertical: verticalPadding ?? 20,
                    ).r,
                    child: Container(
                        decoration: BoxDecoration(
                            borderRadius: const BorderRadius.all(
                                Radius.circular(20),
                            ).w,
                        ),
                        child: child,
                    ),
                ),
            ),
        );
    }
}
```

```
        ),  
        ),  
    );  
}  
}
```

Button: Creates a customizable button with various styles and functionalities to maintain consistency in app design.

```
import 'package:flutter/material.dart';  
import 'package:flutter_screenutil/flutter_screenutil.dart';  
  
class Button extends StatelessWidget {  
    const Button(  
        {super.key,  
        required this.text,  
        this.tap,  
        required this.width,  
        required this.colorBtn,  
        required this.colorTxt,  
        required this.height,  
        this.textSize});  
  
    final String text;  
    final VoidCallback? tap;  
    final double width;  
    final double height;  
    final Color colorBtn;  
    final Color colorTxt;  
    final double? textSize;  
  
    @override  
    Widget build(BuildContext context) {  
        return GestureDetector(  
            onTap: tap,  
            child: Container(  
                decoration: BoxDecoration(  
                    borderRadius: const BorderRadius.all(Radius.circular(23)).w,  
                    color: colorBtn,  
                ),  
                width: width,  
                height: height,  
                child: Center(  
                    child: Text(  
                        text,  
                        style: TextStyle(  
                            color: colorTxt,  
                            fontSize: textSize ?? 16.sp),  
                    ),  
                ),  
            ),  
        );  
    }  
}
```

```

        text,
        style: TextStyle(fontSize: textSize ?? 16.sp, color: colorTxt),
    ),
),
),
);
}
}

```

**Failure Dialogue:** Displays a customizable dialogue for communicating failure states or errors to users in a consistent format.

```

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:on_budget/src/utils/components/button.dart';
import 'package:on_budget/src/utils/constants/colors.dart';

class FailureDialogue extends StatelessWidget {
  const FailureDialogue({
    super.key,
  });

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: const Text('Please check your network connection and try again'),
      actions: [
        Button(
          tap: () => Navigator.pop(context),
          text: 'Okay',
          width: 100.w,
          colorBtn: kPrimaryColor,
          colorTxt: kSecondaryColor,
          height: 50),
        ],
      );
  }
}

```

**Home Background:** Provides a customizable background for the home screen, enhancing visual appeal and branding.

```
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';

import '../helper/constants/colors.dart';

class HomeBackground extends StatelessWidget {
  const HomeBackground({super.key, required this.child});
  final Widget child;
  @override
  Widget build(BuildContext context) {
    return Container(
      width: double.infinity,
      height: double.infinity,
      color: kPrimaryColor,
      child: Column(
        children: [
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              Padding(
                padding: const EdgeInsets.only(left: 20, top: 50).w,
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      'HI Ebram Shereen',
                      style: TextStyle(
                        color: kSecondaryColor,
                        fontSize: 20.sp,
                        fontWeight: FontWeight.w500),
                    ),
                    Text(
                      'Good Morning',
                      style: TextStyle(
                        color: kSecondaryColor,
                        fontSize: 25.sp,
                        fontWeight: FontWeight.w500),
                    ),
                  ],
                ),
              ),
            ],
          ),
        ],
      ),
    );
  }
}
```

```
Padding(
  padding: const EdgeInsets.only(top: 30).w,
  child: Row(
    children: [
      Stack(
        children: [
          IconButton(
            onPressed: () {},
            icon: const Icon(
              Icons.notifications_none,
              color: Colors.white,
              size: 35,
            )),
          Positioned(
            top: ScreenUtil().setHeight(11),
            left: ScreenUtil().setWidth(22),
            child: CircleAvatar(
              backgroundColor: Colors.red,
              radius: 3.r,
            ),
          ),
        ],
      ),
      IconButton(
        onPressed: () {},
        icon: Icon(
          Icons.shopping_cart_outlined,
          color: Colors.white,
          size: ScreenUtil().setWidth(25),
        ),
      ),
      Gap(ScreenUtil().setWidth(20))
    ],
  ),
),
Flexible(
  child: Padding(
    padding: const EdgeInsets.only(top: 10),
    child: Container(
      decoration: BoxDecoration(
        color: const Color(0xFFd6d9da),
        borderRadius: const BorderRadius.only(
          topLeft: Radius.circular(30),

```

```

        topRight: Radius.circular(30),
        ).w),
        child: child,
        ),
        ),
        ],
        )));
}
}

```

**Leading Icon:** Adds a customizable icon to the app bar for easy access to common actions or navigation.

```

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

import '../helper/constants/colors.dart';

class LeadingIcon extends StatelessWidget {
  const LeadingIcon(
    {super.key,
    this.icon,
    this.onPressed,
    this.size,
    this.vertical,
    this.horizontal,
    this.context});
  final IconData? icon;
  final void Function()? onPressed;
  final double? vertical;
  final double? horizontal;
  final double? size;
  final BuildContext? context;

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: EdgeInsets.symmetric(
        vertical: vertical?.toDouble() ?? 0,
        horizontal: horizontal?.toDouble() ?? 0,
      ),
      child: IconButton(
        onPressed: onPressed ??
          () {

```

```

        Navigator.pop(context);
    },
    icon: Icon(
        icon ?? Icons.arrow_back_ios,
        size: size ?? 20.r,
        color: kSecondaryColor,
    )));
}
}
}

```

**Mini Background:** Customizable background for login screens, ensuring a cohesive design throughout the app.

```

import 'package:flutter/material.dart';

import '../helper/constants/colors.dart';

class MiniBackground extends StatelessWidget {
    const MiniBackground({
        super.key,
        required this.child,
        this.width,
        this.height,
    });

    final Widget child;
    final double? width;
    final double? height;

    @override
    Widget build(BuildContext context) {
        return Container(
            color: kPrimaryColor,
            width: width ?? double.infinity,
            height: height ?? double.infinity,
            child: Container(
                decoration: const BoxDecoration(
                    borderRadius: BorderRadius.only(
                        topLeft: Radius.circular(30),
                        topRight: Radius.circular(30),
                    ),
                    color: Color(0xFFFF2F1F),
                ),
                child: child,
            ),
        );
    }
}

```

```
    );
}
}
```

Show Dialogue: A reusable component for displaying customizable dialogues to users, providing information or prompts.

```
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

import '../helper/constants/colors.dart';
import 'button.dart';

Future<dynamic> showDialogWidget(
    BuildContext context, {
    String? firstButtonText,
    String? secondButtonText,
    VoidCallback? firstTap,
    VoidCallback? secondTap,
    String? firstTitle,
    String? secondTitle,
    double? firstTitleSize,
    double? secondTitleSize,
    double? firstButtonSize,
    double? secondButtonSize,
    bool firstButtonVisible = true,
    bool secondButtonVisible = true,
}) {
    return showDialog(
        context: context,
        builder: (context) => AlertDialog(
            title: Column(
                children: [
                    Text(
                        firstTitle ?? '',
                        style: TextStyle(fontSize: firstTitleSize),
                    ),
                    Text(
                        secondTitle ?? '',
                        style: TextStyle(fontSize: secondTitleSize),
                    ),
                ],
            ),
        ),
    );
}
```

```
actions: [
    firstButtonVisible
        ? Visibility(
            visible: true,
            child: Button(
                text: firstButtonText ?? 'sss',
                tap: firstTap,
                width: 100.w,
                colorBtn: kSecondaryColor,
                colorTxt: kCancelText,
                height: 50,
                textSize: firstButtonSize ?? 20,
            ),
        )
        : Visibility(
            visible: false,
            child: Button(
                text: firstButtonText ?? '',
                tap: firstTap,
                width: 100.w,
                colorBtn: kSecondaryColor,
                colorTxt: kCancelText,
                height: 50,
                textSize: firstButtonSize ?? 20,
            ),
        ),
),
Visibility(
    visible: secondButtonVisible,
    child: Button(
        text: secondButtonText ?? '',
        tap: secondTap,
        width: 100.w,
        colorBtn: kPrimaryColor,
        colorTxt: kSecondaryColor,
        height: 50,
        textSize: secondButtonSize ?? 20,
    ),
),
],
),
);
}
```

**Switch:** Creates a customizable switch for toggling settings or options within the app.

```
import 'package:flutter/material.dart';

import '../helper/constants/colors.dart';

Switch switchWidget(
    {required bool value, required void Function(bool)? onChanged}) {
  return Switch(
    activeColor: kSecondaryColor,
    activeTrackColor: ktoogleOn,
    inactiveTrackColor: ktoogleOff,
    inactiveThumbColor: kSecondaryColor,
    splashRadius: 50,
    value: value,
    onChanged: onChanged,
  );
}
```

**TextField:** Allows customization of text input fields with various styling options for consistent appearance.

```
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

import '../helper/constants/colors.dart';

class TextFieldWidget extends StatelessWidget {
  const TextFieldWidget({
    super.key,
    this.hintText,
    this.tap,
    this.onSubmitted,
    this.suffixIcon,
    this.prefixIcon,
    this.prefix,
    this.controller,
    this.text,
  });

  final String? hintText;
  final VoidCallback? tap;
  final void Function(String)? onSubmitted;
```

```
final Widget? suffixIcon;
final Widget? prefixIcon;
final Widget? prefix;
final TextEditingController? controller;
final String? text;

@Override
Widget build(BuildContext context) {
    return TextField(
        onSubmitted: onSubmitted,
        controller: controller,
        decoration: InputDecoration(
            hintText: hintText,
            prefixIcon: prefixIcon,
            prefix: prefix,
            suffixIcon: suffixIcon,
            enabledBorder: OutlineInputBorder(
                borderSide: BorderSide(color: kTextFieldBorder, width: 3.w),
                borderRadius: const BorderRadius.all(
                    Radius.circular(20),
                ).w,
            ),
            focusedBorder: OutlineInputBorder(
                borderSide: BorderSide(color: kPrimaryColor, width: 3.w),
                borderRadius: const BorderRadius.all(
                    Radius.circular(20),
                ),
            ),
            ),
        );
}
```

**TextField**: Similar to **TextField** but tailored specifically for forms, with built-in validation and error handling.

```
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

class TextFormFieldWidget extends StatelessWidget {
  const TextFormFieldWidget({
    super.key,
    required this.labelText,
    required this.keyboardType,
    required this.validate,
    this.obscure = false,
    this.suffixIcon,
    this.paddingTop,
    this.paddingbottom,
    this.controller,
    this.width,
    this.height,
    this.paddingLeft,
    this.paaddingRight,
  });

  final String labelText;
  final TextInputType keyboardType;
  final String? Function(String?) validate;
  final bool obscure;
  final Widget? suffixIcon;
  final double? paddingTop;
  final double? paddingbottom;
  final TextEditingController? controller;
  final double? width;
  final double? height;
  final double? paddingLeft;
  final double? paaddingRight;

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: EdgeInsets.only(
        top: paddingTop ?? 20,
        left: paddingLeft ?? 30,
        right: paaddingRight ?? 30,
        bottom: paddingbottom ?? 20),
      child: SizedBox(
```

```
height: height ?? 50.h,
width: width ?? 400.w,
child: TextFormField(
  controller: controller,
  decoration: InputDecoration(
    suffix: suffixIcon,
    border: const OutlineInputBorder(
      borderRadius: BorderRadius.all(
        Radius.circular(10),
      ),
    ),
    enabledBorder: const OutlineInputBorder(
      borderRadius: BorderRadius.all(
        Radius.circular(10),
      ),
    ),
    disabledBorder: const OutlineInputBorder(
      borderRadius: BorderRadius.all(
        Radius.circular(10),
      ),
    ),
    errorBorder: const OutlineInputBorder(
      borderRadius: BorderRadius.all(
        Radius.circular(10),
      ),
    ),
    labelText: labelText,
  ),
  obscureText: obscure,
  keyboardType: keyboardType,
  validator: validate,
),
),
);
}
}
```

**Waiting Progress Loader:** Displays a customizable loading indicator to inform users of ongoing processes or actions.

```
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:on_budget/src/utils/constants/colors.dart';

class WaitingProgressLoader extends StatefulWidget {
  const WaitingProgressLoader({
    super.key,
  });
  @override
  State<WaitingProgressLoader> createState() => _WaitingProgressLoaderState();
}

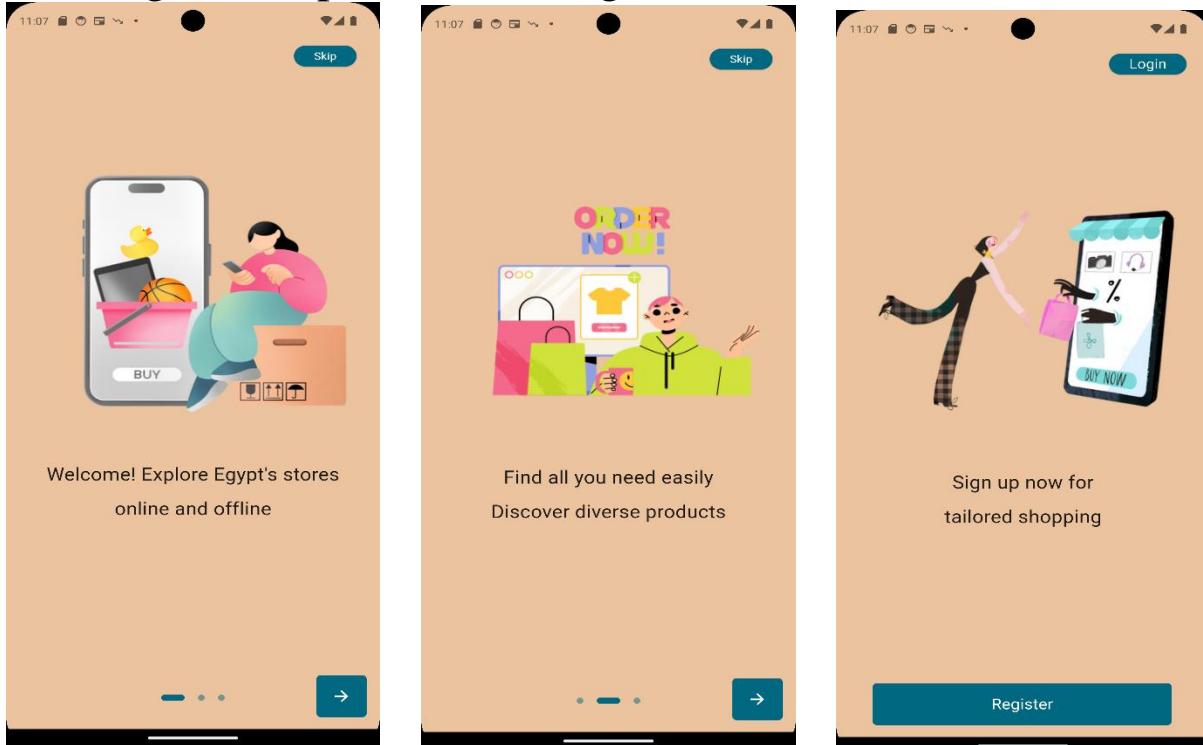
class _WaitingProgressLoaderState extends State<WaitingProgressLoader> {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        width: 150.w,
        height: 150,
        decoration: const BoxDecoration(
          borderRadius: BorderRadius.all(
            Radius.circular(10),
          ),
          color: kProgressLoadingBackground),
        child: const CircularProgressIndicator(
          color: kPrimaryColor,
        )),
    );
  }
}
```

## 5.1.2 OnBoarding screen:

Welcome Screen: Engages users with a friendly greeting and app branding.

Introduction Screens: Educates users about key app features through titles, descriptions, and images.

Call-to-Action Screen: Encourages user action with a persuasive message and a prominent "Register" button.



## Code Implementation onboarding view:

```
import 'package:flutter/material.dart';
import 'package:flutter_onboarding_slider/flutter_onboarding_slider.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

import '../../../../../utils/helper/constants/colors.dart';
import '../../../../../utils/helper/constants/images.dart';
import '../../../../../utils/helper/functions/responsive.dart';
import '../../../../../register/login/presentation/views/login.dart';
import '../../../../../register/signup/modules/signup_options/views/signup_options.dart';
import '../custom_onboarding.dart';

class OnBoardingScreens extends StatelessWidget {
```

```

const OnBoardingScreens({super.key});
static const String id = 'onBoardingScreens';
@Override
Widget build(BuildContext context) {
  return Scaffold(
    body: OnBoardingSlider(
      // finish button =====> that arrow convert to register button
      finishButtonText: 'Register',

      // navigator of finish button
      onFinish: () => Navigator.pushNamed(context, SignupOptions.id),

      // finish button style
      skipIcon: const Icon(
        Icons.arrow_forward,
        color: Colors.white,
      ),

      // background color of upper screen in the (skip/login) button
      headerBackgroundColor: const Color(0xFFEBC49F),

      // background color of bottom screen in the (arrow/register) button
      pageBackgroundColor: const Color(0xFFEBC49F),

      //if true we delete arrow button , slider and we put register button from
      first screen
      hasSkip: true,
      finishButtonStyle: const FinishButtonStyle(
        backgroundColor: kPrimaryColor,
      ),

      // skip button
      skipTextButton: ElevatedButton(
        // navigation.pushnamed() =====> it navigate by id of screen
        //for more info. go to main.dart
        onPressed: () => Navigator.pushNamed(context, SignupOptions.id),
        style: const ButtonStyle(
          backgroundColor: WidgetStatePropertyAll(kPrimaryColor),
        ),
        child: const Text(
          'Skip',
          style: TextStyle(color: Colors.white),
        ),
      ),
    ),
  );
}

```

```
// login button
trailing: ElevatedButton(
    // navigation.pushnamed() =====> it navigate by id of screen
    //for more info. go to main.dart
    onPressed: () => Navigator.pushNamed(context, Login.id),
    style: const ButtonStyle(
        backgroundColor: WidgetStatePropertyAll(kPrimaryColor),
    ),
    child: const Text(
        'Login',
        style: TextStyle(color: Colors.white, fontSize: 18),
    )),
// slider color
controllerColor: kPrimaryColor,
background: [
    Image.asset(
        kOnBoardingOne,
        width: 400.w,
        height: 500.h,
    ),
    Image.asset(
        kOnBoardingTwo,
        width: 400.w,
        height: 500.h,
    ),
    Image.asset(
        kOnBoardingThree,
        width: 380.w,
        height: 500.h,
    ),
],
//speed of background list show in screen
speed: 1.8,
totalPage: 3,
// content of text in every screen
pageBodies: [
    CustomOnBoarding(
        textOne: "Welcome! Explore Egypt's stores",
        textTwo: 'online and offline',
    ),
    CustomOnBoarding(
        textOne: "Find all you need easily",
        textTwo: 'Discover diverse products',
    )
]
```

```

        ),
        CustomOnBoarding(
            textOne: 'Sign up now for',
            textTwo: 'tailored shopping',
            ),
        ],
        );
    );
}
}

```

## onboarding widgets: Custom OnBoarding Widget:

```

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';

// ignore: must_be_immutable
class CustomOnBoarding extends StatelessWidget {
    CustomOnBoarding({
        super.key,
        this.textOne,
        this.textTwo,
        this.textThree,
        this.textOneSize,
        this.textTwoSize,
        this.textThreeSize,
    });
    String id = 'onBoarding';

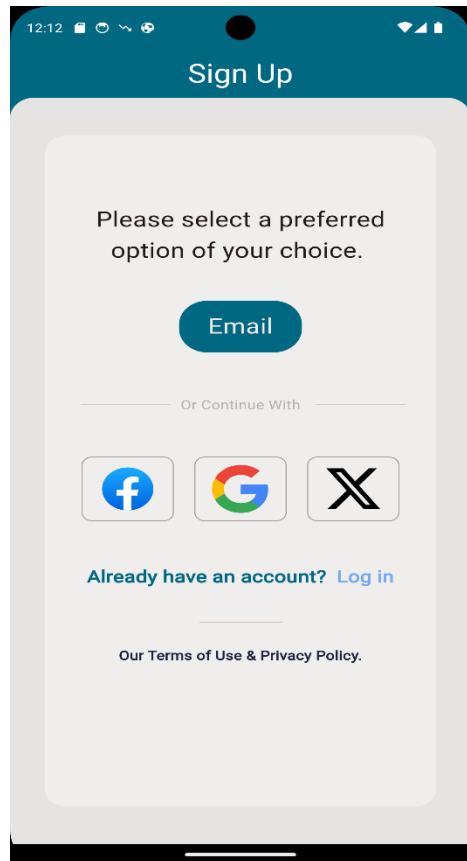
    String? textOne;
    String? textTwo;
    String? textThree;
    double? textOneSize;
    double? textTwoSize;
    double? textThreeSize;
    @override
    Widget build(BuildContext context) {
        return Container(
            padding: const EdgeInsets.all(10),
            alignment: Alignment.center,
            child: Center(

```

```
        child: Column(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [
                SizedBox(
                    height: 430.h,
                ),
                Text(
                    textOne ?? '',
                    style: TextStyle(
                        fontSize: textOneSize ?? 20.sp,
                        fontFamily: 'Merienda',
                    ),
                ),
                Gap(ScreenUtil().setHeight(10)),
                Text(
                    textTwo ?? '',
                    style: TextStyle(
                        fontSize: textTwoSize ?? 20.sp,
                        fontFamily: 'Merienda',
                    ),
                ),
                Gap(ScreenUtil().setHeight(10)),
                Text(
                    textThree ?? '',
                    style: TextStyle(
                        fontSize: textThreeSize ?? 20.sp,
                        fontFamily: 'Merienda',
                    ),
                ),
            ],
        ),
    );
}
```

### 5.1.3 Registration and Login pages: Login page:

Register Options: Offers email and social media registration choices, with a login option available.



### Code Implementation:

```
import 'dart:io' show Platform;

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';
import 'package:text_divider/text_divider.dart';

import '../../../../../utils/components/background.dart';
import '../../../../../utils/components/button.dart';
import '../../../../../utils/helper/auth/google_auth.dart';
import '../../../../../utils/helper/constants/colors.dart';
```

```

import '../../../../../utils/helper/constants/images.dart';
import '../../../../../login/presentation/views/login.dart';
import '../../../../../supplier_or_customer/views/supplier_or_customer.dart';
import '../widgets/signup_social_icons.dart';

class SignupOptions extends StatefulWidget {
  const SignupOptions({super.key});
  static String id = 'SignupOptions';

  @override
  State<SignupOptions> createState() => _SignupOptionsState();
}

class _SignupOptionsState extends State<SignupOptions> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Sign Up'),
      ),
      body: Background(
        child: Padding(
          padding: const EdgeInsets.symmetric(
            horizontal: 10,
            vertical: 20,
          ),
          child: Container(
            decoration: const BoxDecoration(
              color: kSecondaryColor,
              borderRadius: BorderRadius.all(
                Radius.circular(20),
              ),
            ),
            child: Column(
              children: [
                const Gap(80),
                const Text(
                  'Please select a preferred',
                  style: TextStyle(fontSize: 25, fontWeight: FontWeight.w400),
                ),
                const Text(
                  'option of your choice. ',
                  style: TextStyle(fontSize: 25, fontWeight: FontWeight.w400),
                ),
                const Gap(40),
              ],
            ),
          ),
        ),
      ),
    );
  }
}

```

```
        Button(
            colorTxt: kSecondaryColor,
            colorBtn: kPrimaryColor,
            width: 100.w,
            height: 60,
            textSize: 25,
            text: 'Email',
            tap: () =>
                Navigator.pushNamed(context, SupplierOrCustomer.id)),
        const Gap(50),
        Padding(
            padding: const EdgeInsets.symmetric(horizontal: 20),
            child: TextDivider.horizontal(
                text: const Text(
                    'Or Continue With',
                    style: TextStyle(color: kHintColor, fontSize: 15),
                ),
                thickness: .9,
            )));
        const Gap(50),
        Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
                SignupWithSocial(
                    imageurl: kFacebookIcon,
                    width: 75.w,
                    height: 75,
                    tap: () {}),
                const Gap(20),
                SignupWithSocial(
                    imageurl: Platform.isAndroid ? kGoogleIcon : kAppleIcon,
                    width: 75.w,
                    height: 75,
                    tap: () async {
                        signInWithGoogle();
                    }),
                const Gap(20),
                SignupWithSocial(
                    imageurl: kXIIcon,
                    width: 75.w,
                    height: 75,
                    tap: () {}),
            ],
        );
    
```

```

        ],
      ),
      const Gap(50),
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          const Text(
            'Already have an account?',
            style: TextStyle(
              fontSize: 20,
              fontWeight: FontWeight.w600,
              color: kPrimaryColor,
            ),
          ),
          const Gap(10),
          GestureDetector(
            onTap: () => Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => const Login(),
              )),
            child: const Text(
              'Log in',
              style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.w600,
                color: Color(0xFF78AAF4)),
            ),
          ),
        ],
      ),
      const Gap(30),
      const Divider(
        thickness: 1,
        indent: 150,
        endIndent: 150,
      ),
      const Gap(20),
      const Text(
        'Our Terms of Use & Privacy Policy.',
        style: TextStyle(
          fontSize: 15,
          color: Color(0xFF1C2340),
          fontWeight: FontWeight.w600),
      ),
    ),
  ],
),

```

```
        ],
      ),
    ),
  ),
);
}
}
```

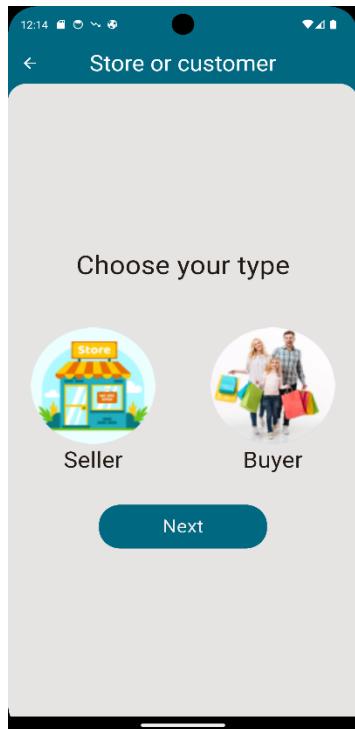
```
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:flutter_svg/svg.dart';

// ignore: must_be_immutable
class SignupWithSocial extends StatelessWidget {
  SignupWithSocial(
    {super.key,
    required this.imageurl,
    required this.width,
    required this.height,
    required this.tap});
  String imageurl;
  double width;
  double height;
  VoidCallback tap;

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: tap,
      child: Container(
        decoration: BoxDecoration(
          borderRadius: const BorderRadius.all(Radius.circular(10)),
          border: Border.all(
            color: Colors.grey,
          )),
        width: width,
        height: height,
        child: Padding(
          padding: const EdgeInsets.all(8),
          child: SvgPicture.asset(
            imageurl,
          ),
        ),
      ),
    );
  }
}
```

```
        ),  
        ),  
    );  
}  
}
```

Register Type: Allows users to select their account type with avatar representations.



## Code Implementation:

```
import 'package:flutter/material.dart';  
import 'package:flutter_screenutil/flutter_screenutil.dart';  
import 'package:gap/gap.dart';  
  
import '../../../../../utils/components/background.dart';  
import '../../../../../utils/components/button.dart';  
import '../../../../../utils/helper/constants/colors.dart';  
import '../customer/presentation/views/signup_for_customer.dart';  
import '../widges/customer_store_choice.dart';  
  
class SupplierOrCustomer extends StatelessWidget {  
    const SupplierOrCustomer({super.key});
```

```

static String id = 'StoreOrCustomer';
@Override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: const Text('Store or customer'),
        ),
        body: Background(
            child: Column(
                children: [
                    const Gap(200),
                    const Text(
                        'Choose your type',
                        style: TextStyle(fontSize: 35),
                    ),
                    const Gap(60),
                    CustomerStoreChoice(),
                    const Gap(40),
                    Button(
                        text: 'Next',
                        tap: () {
                            print(CustomerStoreChoice().customerNavigate);
                            CustomerStoreChoice().customerNavigate == true
                                ? null
                                : Navigator.pushNamed(context, SignUpForCustomer.id);
                        },
                        width: 180.w,
                        colorBtn: kPrimaryColor,
                        colorTxt: kSecondaryColor,
                        height: 60,
                        textSize: 25,
                    )
                ],
            )),
    );
}

import 'package:flutter/material.dart';
import 'package:gap/gap.dart';

import '../../../../../utils/helper/constants/images.dart';
import 'sign_up_choice.dart';

class CustomerStoreChoice extends StatefulWidget {

```

```

CustomerStoreChoice({super.key});
bool? customerNavigate = false;
bool? supplierNavigate = false;
@Override
State<CustomerStoreChoice> createState() => _CustomerStoreChoiceState();
}

class _CustomerStoreChoiceState extends State<CustomerStoreChoice> {
bool isSelectCustomer = false;
bool isSelectStore = false;

@Override
Widget build(BuildContext context) {
return Row(
mainAxisAlignment: MainAxisAlignment.center,
children: [
Column(
children: [
StoreOrCustomer(
image: kCustomerIcon,
tap: () => setState(() {
isSelectCustomer = true;
isSelectStore = false;
widget.supplierNavigate = true;
widget.customerNavigate = false;
}),
isSelect: isSelectCustomer,
),
const Text(
Seller',
style: TextStyle(fontSize: 30),
)
],
),
const Gap(70),
Column(
children: [
StoreOrCustomer(
image: kShopIcon,
tap: () => setState(() {
isSelectCustomer = false;
isSelectStore = true;
widget.customerNavigate = true;
widget.supplierNavigate = false;
}),
)
]
),
]
),
);
}
}

```

```

        isSelect: isSelectStore,
    ),
    const Text(
        'Buyer',
        style: TextStyle(fontSize: 30),
    )
],
),
],
);
}
}

import 'package:flutter/material.dart';

import '../../../../../utils/helper/constants/colors.dart';

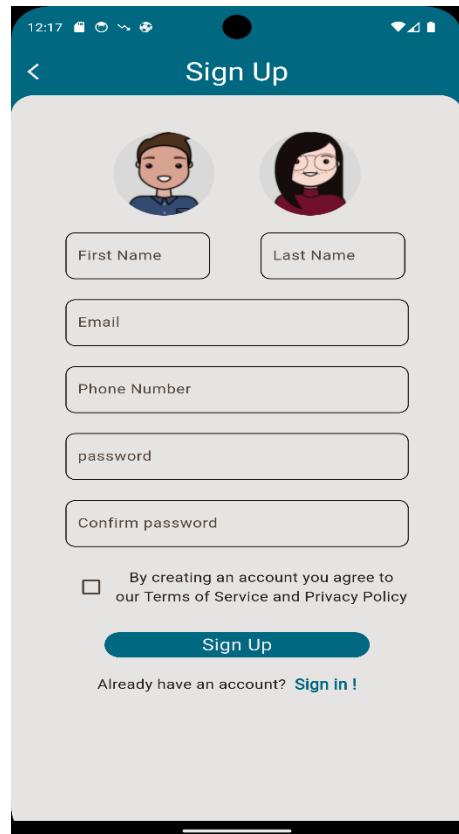
// ignore: must_be_immutable
class StoreOrCustomer extends StatelessWidget {
    StoreOrCustomer(
        {super.key,
        required this.image,
        required this.tap,
        required this.isSelect,
        this.selectedRadius,
        this.unSelectedRadius,
        this.selectedRadiusColor,this.isCustomer,this.isSupplier});
    bool isSelect = false;
    final String image;
    final VoidCallback tap;
    final double? selectedRadius;
    final double? selectedRadiusColor;
    final double? unSelectedRadius;
    bool? isCustomer = false;
    bool? isSupplier = false;

    @override
    Widget build(BuildContext context) {
        return isSelect == true
            ? CircleAvatar(
                radius: selectedRadiusColor ?? 83,
                backgroundColor: kPrimaryColor,
                child: GestureDetector(
                    onTap: tap,
                    child: CircleAvatar(

```

```
        radius: selectedRadius ?? 80,
        backgroundColor: const Color(0xffD9D9D9),
        backgroundImage: AssetImage(image),
      ),
    ),
  )
: GestureDetector(
  onTap: tap,
  child: CircleAvatar(
    radius: unSelectedRadius ?? 80,
    backgroundColor: const Color(0xffD9D9D9),
    backgroundImage: AssetImage(image),
  ),
);
}
}
```

**Register by Email (Customer):** Enables customers to register with gender selection and standard personal details.



## Code Implementation:

```
class CustomerModel {  
    String? firstName;  
    String? lastName;  
    String? handle;  
    String? phoneNumber;  
    String? password;  
    String? gender;  
    String? address;  
  
    CustomerModel({  
        required this.firstName,  
        required this.lastName,  
        required this.handle,  
        required this.phoneNumber,  
        required this.password,  
        required this.gender,  
        required this.address,  
    });  
  
    factory CustomerModel.fromJson(Map<String, dynamic> json) {  
        return CustomerModel(  
            firstName: json['firstName'],  
            lastName: json['lastName'],  
            handle: json['handle'],  
            phoneNumber: json['phoneNumber'],  
            password: json['password'],  
            gender: json['gender'],  
            address: json['address'],  
        );  
    }  
}  
  
import  
'package:on_budget/src/modules/register/signup/modules/customer/data/models/customer_model.dart';  
import  
'package:on_budget/src/modules/register/signup/modules/customer/data/service/customer_service.dart';  
import 'package:on_budget/src/utils/helper/constants/api.dart';  
  
class CustomerRepository {
```

```

final CustomerService customerService;

CustomerRepository(this.customerService);

Future<CustomerModel> customerRepository({
    required Map<String, String> customerData,
}) async {
    await customerService.post(
        url: RegisterApi.customerSignUp,
        customerData: customerData,
    );
    return CustomerModel(
        firstName: customerData['firstName']!,
        lastName: customerData['lastName']!,
        handle: customerData['handle']!,
        phoneNumber: customerData['phoneNumber']!,
        password: customerData['password']!,
        gender: customerData['gender']!,
        address: customerData['address'] ?? '',
    );
}
}

import 'dart:convert';
import 'package:http/http.dart' as http;

class CustomerService {
    Future<int> post({
        required String url,
        required Map<String, String> customerData,
    }) async {
        try {
            final response = await http.post(
                Uri.parse(url),
                body: jsonEncode(customerData),
                headers: {
                    "Content-Type": "application/json",
                    "Accept": "application/json"
                },
            );

            if (response.statusCode == 201) {
                return int.parse(response.body);
            } else {
                throw Exception('Failed to post customer data: ${response.statusCode}');
            }
        }
    }
}

```

```

        }
    } catch (e) {
        throw Exception('Failed to post customer data: $e');
    }
}

import 'dart:developer';

import 'package:bloc/bloc.dart';

import '../data/repository/customer_repository.dart';
import 'customer_state.dart';

class CustomerCubit extends Cubit<CustomerStates> {
    final CustomerRepository customerRepository;

    CustomerCubit(this.customerRepository) : super(CustomerInitial());

    Future<void> customerCubit(
        {required Map<String, String> customerData}) async {
        emit(CustomerWaiting());
        try {
            final result = await customerRepository.customerRepository(
                customerData: customerData);
            log('response of cubit $result');
            emit(CustomerSuccess(customerModel: result));
        } catch (e) {
            log('exception of cubit ${e.toString()}');
            emit(CustomerFailure(e.toString()));
        }
    }
}

import '../data/models/customer_model.dart';

abstract class CustomerStates {}

class CustomerInitial extends CustomerStates {}

class CustomerWaiting extends CustomerStates {}

class CustomerSuccess extends CustomerStates {
    final CustomerModel customerModel;
}

```

```

        CustomerSuccess({required this.customerModel});
    }

    class CustomerFailure extends CustomerStates {
        final String error;

        CustomerFailure(this.error);
    }

    import 'package:flutter/material.dart';
    import 'package:flutter_bloc/flutter_bloc.dart';
    import 'package:on_budget/src/utils/components/button.dart';
    import 'package:on_budget/src/utils/components/failure_dialogue.dart';
    import 'package:on_budget/src/utils/components/waiting_progress_loader.dart';
    import 'package:on_budget/src/utils/helper/constants/colors.dart';
    import 'package:simple_circular_progress_bar/simple_circular_progress_bar.dart';
    import '../../../../../customer_modules/home/presentation/views/home.dart';
    import '../../../../../data/repository/customer_repository.dart';
    import '../../../../../data/service/customer_service.dart';
    import '../../../../../logic/customer_cubit.dart';
    import '../../../../../logic/customer_state.dart';
    import '../../../../../account_created/views/account_created.dart';
    import '../widgets/sign_up_form_for_customer.dart';
    import '../../../../../utils/components/leading_icon.dart';

    class SignUpForCustomer extends StatefulWidget {
        const SignUpForCustomer({super.key});
        static String id = 'SignUpWithEmailForCustomer';

        @override
        State<SignUpForCustomer> createState() => _SignUpForCustomerState();
    }

    GlobalKey<FormState> formKey = GlobalKey();

    class _SignUpForCustomerState extends State<SignUpForCustomer> {
        @override
        Widget build(BuildContext context) {
            return BlocProvider(
                create: (context) => CustomerCubit(CustomerRepository(CustomerService())),
                child: Scaffold(
                    appBar: AppBar(
                        title: const Text('Sign Up'),
                        leading: const LeadingIcon(),
                ),

```

```

        body: BlocConsumer<CustomerCubit, CustomerStates>(
            listener: (context, state) {
                if (state is CustomerWaiting) {
                    const WaitingProgressLoader();
                } else if (state is CustomerSuccess) {
                    Navigator.pushNamed(context, AccountCreated.id);
                } else if (state is CustomerFailure) {
                    showDialog(
                        context: context,
                        builder: (context) => const FailureDialouge());
                }
            },
            builder: (context, state) {
                return SignUpFormforCustomer();
            },
        ),
    );
}

import 'package:flutter/material.dart';
import 'package:gap/gap.dart';

import '../../../../../utils/helper/constants/images.dart';
import '../../../../../supplier_or_customer/widges/sign_up_choice.dart';

class GenderChoice extends StatefulWidget {
    GenderChoice({Key? key}) : super(key: key);
    String? gender;

    @override
    GenderChoiceState createState() => GenderChoiceState();
}

class GenderChoiceState extends State<GenderChoice> {
    bool isSelectedMan = false;
    bool isSelectedWoman = false;

    @override
    Widget build(BuildContext context) {
        return Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
                StoreOrCustomer(

```

```

        image: kBoyIcon,
        tap: () => setState(() {
            isSelectMan = true;
            isSelectWoman = false;
            widget.gender = "man";
        }),
        isSelect: isSelectMan,
        selectedRadius: 50,
        selectedRadiusColor: 52,
        unSelectedRadius: 50,
    ),
    const Gap(45),
    StoreOrCustomer(
        image: kGirlIcon,
        tap: () => setState(() {
            isSelectMan = false;
            isSelectWoman = true;
            widget.gender = "woman";
        }),
        isSelect: isSelectWoman,
        selectedRadius: 50,
        selectedRadiusColor: 52,
        unSelectedRadius: 50,
    ),
],
);
}
}

import 'dart:developer';

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';
import '../../../../../custom_widgets/signup_alert_dialog.dart';
import '../../../../../logic/customer_cubit.dart';
import '../../../../../custom_widgets/already_have_account.dart';
import '../../../../../custom_widgets/confirm_policy.dart';
import 'gender_choice.dart';
import 'signup_textfield_for_customer.dart';
import '../../../../../utils/components/button.dart';
import '../../../../../utils/helper/constants/colors.dart';
import '../../../../../utils/components/background.dart';

```

```

class SignUpFormForCustomer extends StatelessWidget {
  SignUpFormForCustomer({Key? key}) : super(key: key);

  final GlobalKey<FormState> formKey = GlobalKey<FormState>();
  final TextEditingController firstNameController = TextEditingController();
  final TextEditingController lastNameController = TextEditingController();
  final TextEditingController phoneController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  final TextEditingController emailController = TextEditingController();

  // Create a single instance of GenderChoice with a GlobalKey
  final GlobalKey<GenderChoiceState> genderChoiceKey =
    GlobalKey<GenderChoiceState>();
  final GlobalKey<ConfirmPolicyState> confirmPolicykey =
    GlobalKey<ConfirmPolicyState>();

  bool genderisSelected = false;
  bool isValid = false;
  @override
  Widget build(BuildContext context) {
    return Background(
      child: ListView(
        physics: const BouncingScrollPhysics(),
        children: [
          Column(
            children: [
              const Gap(20),
              GenderChoice(key: genderChoiceKey), // Use the key here
              const Gap(20),
              SignupTextFieldForCustomer(
                formKey: formKey,
                firstNameController: firstNameController,
                lastNameController: lastNameController,
                phoneController: phoneController,
                passwordController: passwordController,
                emailController: emailController,
              ),
              const Gap(20),
              ConfirmPolicy(
                key: confirmPolicykey,
              ),
              const Gap(30),
              Button(
                text: 'Sign Up',
                tap: () {

```

```

        if (formKey.currentState!.validate()) {
            formKey.currentState!.save();
            isValid = true;
        }
        if (isValid) {
            final cubit = context.read<CustomerCubit>();
            final genderState = genderChoiceKey.currentState;
            final isGenderSelected = genderState?.widget.gender;
            final isConfirmPolicySelected =
                confirmPolicykey.currentState?.checkBox ?? false;
            if (isGenderSelected == null) {
                showDialog(
                    context: context,
                    builder: (context) {
                        return const SignUpAlertDialog(
                            text: 'Please choose your gender type',
                            button: 'Okaaay',
                        );
                    },
                );
            } else if (isConfirmPolicySelected == false) {
                showDialog(
                    context: context,
                    builder: (context) {
                        return const SignUpAlertDialog(
                            text: 'Accept to our Terms and Policies',
                            button: 'Okaaay',
                        );
                    },
                );
            } else if (isGenderSelected == 'man' ||
                isGenderSelected == 'woman' ||
                isConfirmPolicySelected == true ||
                isValid == true) {
                log('$isValid');
                cubit.customerCubit(customerData: {
                    "firstName": firstNameController.text.toLowerCase(),
                    "lastName": lastNameController.text.toLowerCase(),
                    "handle": emailController.text.toLowerCase(),
                    "gender": isGenderSelected.toLowerCase(),
                    "phoneNumber": phoneController.text,
                    "address": '',
                    "password": passwordController.text
                });
            }
        }
    }
}

```

```

        },
    ],
    width: 220.w,
    colorBtn: kPrimaryColor,
    colorTxt: kSecondaryColor,
    height: 30,
    textSize: 20,
),
const Gap(20),
const AlreadyHaveAccount(),
],
),
],
),
);
}
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';
import
'../../../../login/presentation/widgets/forget_password/phone_verification_textfield.dart';
import 'gender_choice.dart';

import '../../../../../utils/components/text_form_field.dart';
import '../../../../../utils/helper/functions/regex.dart';

class SignupTextFieldForCustomer extends StatefulWidget {
  const SignupTextFieldForCustomer({
    super.key,
    required this.formKey,
    required this.firstNameController,
    required this.lastNameController,
    required this.phoneController,
    required this.passwordController,
    required this.emailController,
    // required this.password,
    // required this.username,
    // required this.firstName,
    // required this.lastName,
    // required this.phone
  });
}

```

```

final GlobalKey<FormState> formKey;
final TextEditingController firstNameController;
final TextEditingController lastNameController;
final TextEditingController phoneController;
final TextEditingController passwordController;
final TextEditingController emailController;

@Override
State<SignupTextFieldForCustomer> createState() =>
    _SignupTextFieldForCustomerState();
}

const bool hide = false;

class _SignupTextFieldForCustomerState
    extends State<SignupTextFieldForCustomer> {
@Override
Widget build(BuildContext context) {
    return Form(
        key: widget.formKey,
        child: Column(
            children: [
                Row(
                    children: [
                        //first name
                        TextFormFieldWidget(
                            controller: widget.firstNameController,
                            width: 120.w,
                            paddingRight: 0,
                            paddingLeft: 30,
                            paddingBottom: 0,
                            paddingTop: 0,
                            labelText: 'First Name',
                            keyboardType: TextInputType.name,
                            validate: (value) {
                                if (value!.isNotEmpty) {
                                    if (!value.contains(firstNameRegExp)) {
                                        return 'Please enter a valid first name';
                                    }
                                } else if (value.isEmpty) {
                                    return 'This field is empty!!!';
                                }
                                // widget.firstName = value;

                                return null;
                            }
                        )
                    ],
                )
            ],
        )
    );
}
}

```

```
        },
    ),
    //last name
    const Gap(30),
    TextFormFieldWidget(
        controller: widget.lastNameController,
        width: 120.w,
        paddingLeft: 20,
        paddingTop: 0,
        paddingBottom: 0,
        labelText: 'Last Name',
        keyboardType: TextInputType.name,
        validate: (value) {
            if (value!.isEmpty) {
                if (!value.contains(lastNameRegExp)) {
                    return 'Please enter a valid last name';
                }
            } else if (value.isEmpty) {
                return 'This field is empty!!!';
            }
            // widget.lastName = value;
            return null;
        },
    ),
    ],
),
const Gap(20),
//email
TextFormFieldWidget(
    controller: widget.emailController,
    paddingBottom: 0,
    paddingTop: 0,
    labelText: 'Email',
    keyboardType: TextInputType.emailAddress,
    validate: (value) {
        if (value!.isEmpty) {
            return 'Email is required';
        } else if (value.toLowerCase().isEmpty) {
            if (!value.contains(emailRegExp)) {
                return 'Please enter a valid email address';
            }
        }
        // widget.email = value;
        return null;
    },
)
```

```
        },
    ),
    const Gap(20),
    // phone number

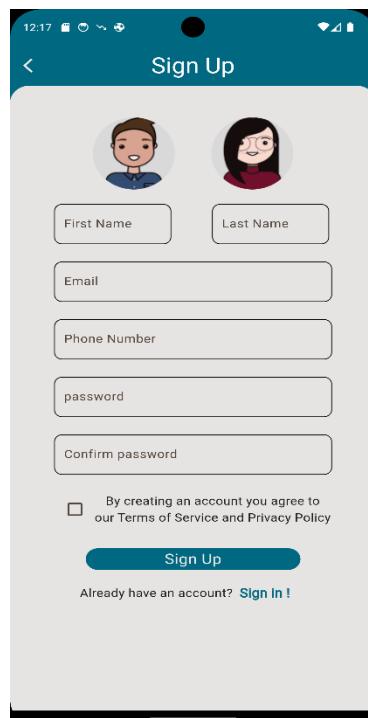
    TextFormFieldWidget(
        controller: widget.phoneController,
        paddingbottom: 0,
        paddingTop: 0,
        labelText: 'Phone Number',
        keyboardType: TextInputType.emailAddress,
        validate: (value) {
            if (value!.isEmpty) {
                return 'Phone number is required';
            } else if (value.toLowerCase().isNotEmpty) {
                if (!value.contains(phoneRegExp)) {
                    return 'Please enter a valid Phone number';
                }
            }
            // widget.email = value;
            return null;
        },
    ),
    const Gap(20),
    //password
    TextFormFieldWidget(
        paddingbottom: 0,
        paddingTop: 0,
        controller: widget.passwordController,
        labelText: 'password',
        obscure: hide,
        keyboardType: TextInputType.emailAddress,
        validate: (value) {
            // widget.password = value;
            if (value!.isEmpty) {
                return 'password is required';
            } else if (value.isNotEmpty) {
                if (!value.contains(passwordRegExp)) {
                    return 'Password must be at least contain \n1. One small character \n2. One capital character \n3. One special character \n4. One number';
                }
            }
            // widget.password = value;
            return null;
        },
    ),
```

```

),
const Gap(20),
TextFormFieldWidget(
  paddingTop: 0,
  paddingBottom: 0,
  labelText: 'Confirm password',
  keyboardType: TextInputType.emailAddress,
  obscure: hide,
  validate: (value) {
    if (value != widget.passwordController.text || value!.isEmpty) {
      return 'Password and Confirm Password must be same';
    }
    return null;
  },
),
],
),
);
}
}
}

```

**Register by Email (Supplier):** Collects supplier-specific details along with standard registration information.



## Code Implementation:

```
class SupplierModel {  
    String? firstName;  
    String? lastName;  
    String? handle;  
    String? phoneNumber;  
    String? password;  
    String? companyName;  
  
    SupplierModel(  
        {this.firstName,  
         this.lastName,  
         this.handle,  
         this.phoneNumber,  
         this.password,  
         this.companyName});  
  
    SupplierModel.fromJson(Map<String, dynamic> json) {  
        firstName = json['firstName'];  
        lastName = json['lastName'];  
        handle = json['handle'];  
        phoneNumber = json['phoneNumber'];  
        password = json['password'];  
        companyName = json['companyName'];  
    }  
}  
  
import 'dart:convert';  
  
import 'package:http/http.dart' as http;  
  
class SupplierService {  
    Future<Map<String, String>> supplierService({  
        required String url,  
        required Map<String, String> supplierData,  
    }) async {  
        try {  
            final response = await http  
                .post(Uri.parse(url), body: jsonEncode(supplierData), headers: {  
                    "Content-Type": "application/json",  
                    "Accept": "application/json",  
                });  
            return jsonDecode(response.body);  
        } on Exception catch (e) {  
    }  
}
```

```

        throw Exception('Failed to post supplier data $e');
    }
}
}

import '../models/supplier_model.dart';
import '../service/supplier_service.dart';
import '../../../../../utils/helper/constants/api.dart';

class SupplierRepository {
    SupplierService supplierService;
    SupplierRepository({
        required this.supplierService,
    });
    Future<SupplierModel> supplierRepository(
        Map<String, String> supplierData) async {
        final response = await supplierService.supplierService(
            url: RegisterApi.supplierSignUp, supplierData: supplierData);

        return SupplierModel.fromJson(response);
    }
}

import 'package:flutter_bloc/flutter_bloc.dart';
import '../data/repository/supplier_repository.dart';
import 'add_supplier_states.dart';

class SupplierCubit extends Cubit<SupplierStates> {
    final SupplierRepository supplierRepository;
    SupplierCubit({required this.supplierRepository}) : super(SupplierInitial());
    Future<void> supplierCubit({required Map<String, String> supplierData}) async {
        emit(SupplierWaiting());
        try {
            final result = await supplierRepository.supplierRepository(supplierData);
            emit(SupplierSuccess(supplierModel: result));
        } catch (e) {
            emit(
                SupplierFailure(
                    exception: Exception(
                        'there was an exception and the exception code is
${e.toString()}'),
                ),
            );
        }
    }
}

```

```

}

// ignore_for_file: public_member_api_docs, sort_constructors_first
import
'package:on_budget/src/modules/register/signup/modules/supplier/data/models/supplier_model.dart';

class SupplierStates {}

class SupplierInitial extends SupplierStates {}

class SupplierWaiting extends SupplierStates {}

class SupplierSuccess extends SupplierStates {
  SupplierModel supplierModel;
  SupplierSuccess({
    required this.supplierModel,
  });
}

class SupplierFailure extends SupplierStates {
  final Exception exception;

  SupplierFailure({required this.exception});
}

// ignore_for_file: public_member_api_docs, sort_constructors_first
import
'package:on_budget/src/modules/register/signup/modules/supplier/data/models/supplier_model.dart';

class SupplierStates {}

class SupplierInitial extends SupplierStates {}

class SupplierWaiting extends SupplierStates {}

class SupplierSuccess extends SupplierStates {
  SupplierModel supplierModel;
  SupplierSuccess({
    required this.supplierModel,
  });
}

class SupplierFailure extends SupplierStates {

```

```

final Exception exception;

SupplierFailure({required this.exception});
}

import 'package:flutter_bloc/flutter_bloc.dart';
import '../data/repository/supplier_repository.dart';
import 'add_supplier_states.dart';

class SupplierCubit extends Cubit<SupplierStates> {
    final SupplierRepository supplierRepository;
    SupplierCubit({required this.supplierRepository}) : super(SupplierInitial());
    Future<void> supplierCubit({required Map<String, String> supplierData}) async {
        emit(SupplierWaiting());
        try {
            final result = await supplierRepository.supplierRepository(supplierData);
            emit(SupplierSuccess(supplierModel: result));
        } catch (e) {
            emit(
                SupplierFailure(
                    exception: Exception(
                        'there was an exception and the exception code is
${e.toString()}'),
                    ),
            );
        }
    }
}

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';

import '../../../../../data/repository/supplier_repository.dart';
import '../../../../../data/service/supplier_service.dart';
import '../../../../../logic/add_supplier_cubit.dart';
import '../../../../../logic/add_supplier_states.dart';
import '../widgets/sign_up_form_for_supplier.dart';

class SignUpForSupplier extends StatefulWidget {
    const SignUpForSupplier({super.key});
    static String id = 'SignUpForSupplier';

    @override
    State<SignUpForSupplier> createState() => _SignUpWithEmailState();
}

```

```

class _SignUpWithEmailState extends State<SignUpForSupplier> {
  @override
  Widget build(BuildContext context) {
    return BlocProvider(
      create: (context) => SupplierCubit(
        supplierRepository:
            SupplierRepository(supplierService: SupplierService())),
      child: Scaffold(
        appBar: AppBar(
          title: const Text('Sign Up'),
        ),
        body: BlocConsumer<SupplierCubit, SupplierStates>(
          listener: (context, state) {
            if (state is SupplierWaiting) {
              ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(
                  content: Text('Loading...'),
                ),
              );
            } else if (state is SupplierFailure) {
              ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(
                  content: Text(state.exception.toString()),
                ),
              );
            } else if (state is SupplierWaiting) {
              ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(
                  content: Text('success'),
                ),
              );
            }
          },
          builder: (context, state) {
            return SignUpFormforSupplier();
          },
        ),
      ),
    );
  }

  import 'package:flutter/material.dart';
  import 'package:flutter_bloc/flutter_bloc.dart';

```

```

import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';

import '../../../../../utils/components/background.dart';
import '../../../../../utils/components/button.dart';
import '../../../../../utils/helper/constants/colors.dart';
import '../../../../../custom_widgets/already_have_account.dart';
import '../../../../../custom_widgets/confirm_policy.dart';
import '../../../../../custom_widgets/signup_alert_dialog.dart';
import '../../../../../logic/add_supplier_cubit.dart';
import 'signup_textfield_for_supplier.dart';

class SignUpFormforSupplier extends StatelessWidget {
  SignUpFormforSupplier({super.key});

  final GlobalKey<FormState> signupForSupplierFormKey = GlobalKey();
  final GlobalKey<ConfirmPolicyState> confirmPolicyKey =
    GlobalKey<ConfirmPolicyState>();
  final TextEditingController firstNameController = TextEditingController();
  final TextEditingController lastNameController = TextEditingController();
  final TextEditingController phoneController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  final TextEditingController emailController = TextEditingController();
  final TextEditingController companyNameController = TextEditingController();
  bool isValid = false;

  @override
  Widget build(BuildContext context) {
    return Background(
      child: ListView(
        physics: const BouncingScrollPhysics(),
        children: [
          Column(
            children: [
              const Gap(20),
              SignupTextFieldForSupplier(
                companyNameController: companyNameController,
                formKey: signupForSupplierFormKey,
                firstNameController: firstNameController,
                lastNameController: lastNameController,
                // phoneController: phoneController,
                passwordController: passwordController,
                emailController: emailController,
              ),
              const Gap(20),
            ],
          ),
        ],
      ),
    );
  }
}

```

```

        ConfirmPolicy(key: confirmPolicyKey),
        const Gap(30),
        Button(
            text: 'Sign Up',
            tap: () {
                if (signupForSupplierFormKey.currentState!.validate()) {
                    signupForSupplierFormKey.currentState!.save();
                    isValid = true;
                }
                if (isValid) {
                    final isConfirmPolicySelected =
                        confirmPolicyKey.currentState?.checkBox ?? false;
                    if (isConfirmPolicySelected) {
                        final cubit = context.read<SupplierCubit>();
                        cubit.supplierCubit(supplierData: {
                            'firstName': firstNameController.text,
                            'lastName': lastNameController.text,
                            'phone': phoneController.text,
                            'password': passwordController.text,
                            'handle': emailController.text,
                            'companyName': companyNameController.text,
                        });
                    } else {
                        showDialog(
                            context: context,
                            builder: (context) {
                                return const SignUpAlertDialog(
                                    text: 'Accept to our Terms and Policies',
                                    button: 'Okaaay',
                                );
                            },
                        );
                    }
                }
            },
            width: 220.w,
            colorBtn: kPrimaryColor,
            colorTxt: kSecondaryColor,
            height: 30,
            textSize: 20,
        ),
        const Gap(20),
        const AlreadyHaveAccount(),
    ],
),

```

```

        ],
    ),
);
}
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

import '../../../../../utils/components/text_form_field.dart';
import '../../../../../utils/functions/regex.dart';

class SignupTextFieldForSupplier extends StatefulWidget {
  const SignupTextFieldForSupplier({
    super.key,
    required this.formKey,
    required this.firstNameController,
    required this.lastNameController,
    // required this.phoneController,
    required this.passwordController,
    required this.emailController,
    required this.companyNameController,
    // required this.password,
    // required this.username,
    // required this.firstName,
    // required this.lastName,
    // required this.phone
  });
  final GlobalKey<FormState> formKey;

  final TextEditingController firstNameController;
  final TextEditingController lastNameController;
  // final TextEditingController phoneController;
  final TextEditingController passwordController;
  final TextEditingController emailController;
  final TextEditingController companyNameController;

  @override
  State<SignupTextFieldForSupplier> createState() =>
      _SignupTextFieldForSupplierState();
}

const bool hide = false;

```

```

class _SignupTextFieldForSupplierState
    extends State<SignupTextFieldForSupplier> {
@override
Widget build(BuildContext context) {
return Form(
key: widget.formKey,
child: Column(
children: [
Row(
children: [
//first name
TextFormFieldWidget(
controller: widget.firstNameController,
width: 150.w,
paddingRight: 0,
paddingLeft: 30,
paddingBottom: 0,
paddingTop: 0,
labelText: 'First Name',
keyboardType: TextInputType.name,
validate: (value) {
if (value!.isNotEmpty) {
if (!value.contains(firstNameRegExp)) {
return 'Please enter a valid first name';
}
} else if (value.isEmpty) {
return 'This field is empty!!!';
}
// widget.firstName = value;

return null;
},
),
//last name
TextFormFieldWidget(
controller: widget.lastNameController,
width: 150.w,
paddingLeft: 46,
paddingTop: 0,
paddingBottom: 0,
labelText: 'Last Name',
keyboardType: TextInputType.name,
validate: (value) {
if (value!.isNotEmpty) {
if (!value.contains(lastNameRegExp)) {

```

```

                return 'Please enter a valid last name';
            }
        } else if (value.isEmpty) {
            return 'This field is empty!!!';
        }
        // widget.lastName = value;
        return null;
    },
),
],
),
),
TextFormFieldWidget(
    controller: widget.companyNameController,
    width: 150.w,
    paddingRight: 0,
    paddingLeft: 30,
    paddingBottom: 0,
    paddingTop: 0,
    labelText: 'Company Name',
    keyboardType: TextInputType.name,
    validate: (value) {
        if (value!.isEmpty) {
            if (!value.contains(firstNameRegExp)) {
                return 'Please enter a valid company name';
            }
        } else if (value.isEmpty) {
            return 'This field is empty!!!';
        }
        // widget.firstName = value;

        return null;
    },
),
//email
TextFormFieldWidget(
    controller: widget.emailController,
    paddingBottom: 0,
    paddingTop: 0,
    labelText: 'Email',
    keyboardType: TextInputType.emailAddress,
    validate: (value) {
        if (value!.isEmpty) {
            return 'Email is required';
        } else if (value.toLowerCase().isNotEmpty) {
            if (!value.contains(emailRegExp)) {

```

```

                return 'Please enter a valid email address';
            }
        }
        // widget.email = value;
        return null;
    },
),

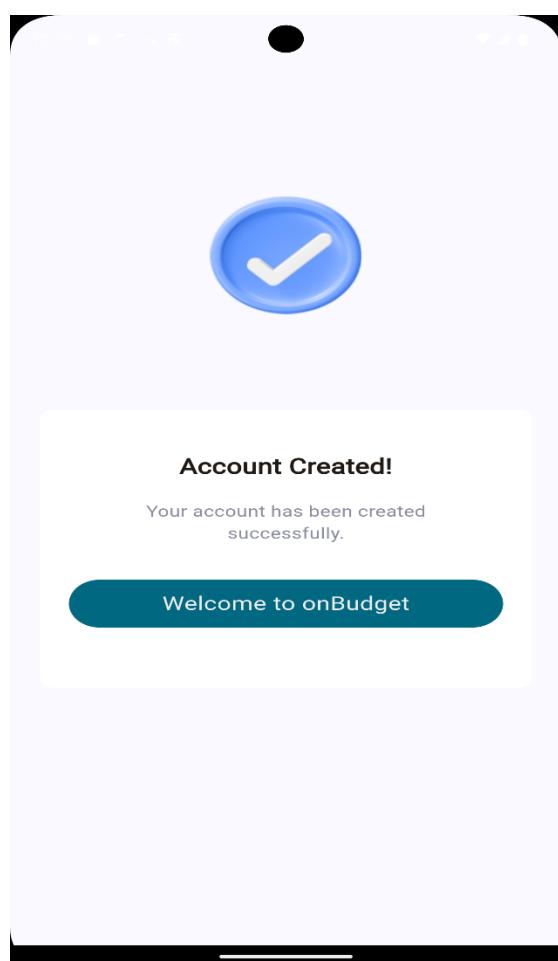
// phone number
// PhoneVerificationTextField(
//   phoneNumber: '01211212239',
//   lengthCheck: true,
//   controller: widget.phoneController,
// ),
//password
TextFormFieldWidget(
    paddingbottom: 0,
    paddingTop: 0,
    controller: widget.passwordController,
    labelText: 'password',
    obscure: hide,
    keyboardType: TextInputType.emailAddress,
    validate: (value) {
        // widget.password = value;
        if (value!.isEmpty) {
            return 'password is required';
        } else if (value.isNotEmpty) {
            if (!value.contains(passwordRegExp)) {
                return 'Password must be at least contain \n1. One small
character \n2. One capital character \n3. One special character \n4. One number';
            }
        }
        // widget.password = value;
        return null;
    },
),

//confirm password
TextFormFieldWidget(
    paddingTop: 0,
    paddingbottom: 0,
    labelText: 'Confirm password',
    keyboardType: TextInputType.emailAddress,
    obscure: hide,

```

```
validate: (value) {
    if (value!.isEmpty) {
        return 'check password is required';
    } else if (value.isNotEmpty) {
        if (value != widget.passwordController.text) {
            return '';
        }
    }
    return null;
},
),
],
),
);
}
}
```

Account Created: when account successfully created.



## Code Implementation:

```
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';

import '../widgets/account_created_widget.dart';

class AccountCreated extends StatelessWidget {
    const AccountCreated({super.key});
    static String id = 'AccountCreated';
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Container(
                color: const Color(0xFFFF9F9F),
                child: Column(
                    children: [
                        const Gap(180),
                        Center(
                            child: SizedBox(
                                width: 139.w,
                                height: 143,
                                child: Image.asset(
                                    'lib/src/res/images/register/account_created/account_create
d.jpg')),
                        ),
                        const Gap(90),
                        const AccountCreatedWidget()
                    ],
                ),
            ),
        );
    }
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';

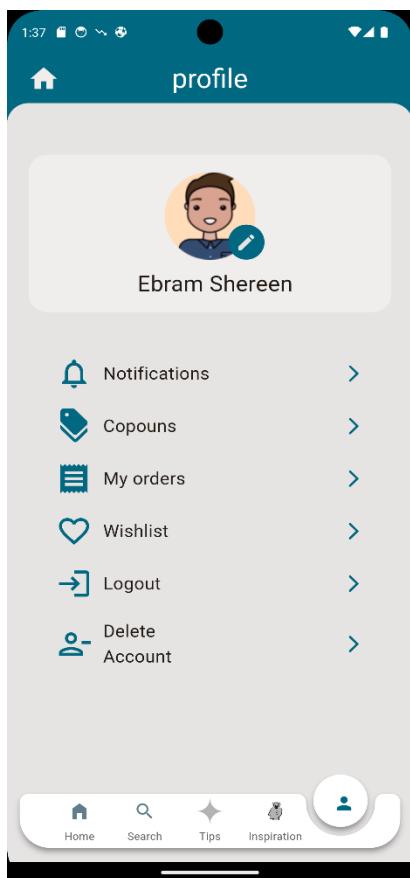
import '../../../../../utils/components/button.dart';
import '../../../../../utils/helper/constants/colors.dart';

class AccountCreatedWidget extends StatelessWidget {
    const AccountCreatedWidget({Key? key}) : super(key: key);
```

```
@override
Widget build(BuildContext context) {
  return Container(
    width: 335.w,
    height: 291,
    decoration: BoxDecoration(
      color: Colors.white,
      borderRadius: BorderRadius.circular(10),
    ),
    child: Column(
      children: [
        const Gap(44),
        const Text(
          'Account Created!',
          style: TextStyle(
            fontSize: 22,
            fontWeight: FontWeight.w700,
          ),
        ),
        const Gap(20),
        const Text(
          'Your account has been created',
          style: TextStyle(
            color: Color(0xFF8A8D9F),
            fontSize: 16,
            fontWeight: FontWeight.w400,
          ),
        ),
        const Text(
          'successfully.',
          style: TextStyle(
            color: Color(0xFF8A8D9F),
            fontSize: 16,
            fontWeight: FontWeight.w400,
          ),
        ),
        const Gap(37),
        Button(
          height: 50,
          colorTxt: kSecondaryColor,
          text: 'Welcome to onBudget',
          tap: () {},
          width: 295.w,
          colorBtn: kPrimaryColor,
        ),
      ],
    ),
  );
}
```

```
        textSize: 20,
    ),
),
],
),
);
}
}
```

## 5.1.4 Profile Screen:



## Code Implementation:

```
import 'package:flutter/material.dart';
import 'package:gap/gap.dart';

import '../../../../../utils/components/background.dart';
import '../../../../../utils/components/bottom_bar/bottom_navbar.dart';
import '../../../../../utils/components/leading_icon.dart';
```

```

import '../../../../../utils/components/show_dialog.dart';
import '../../../../../customer_modules/home/presentation/views/home.dart';
import '../../../../../register/login/presentation/views/login.dart';
import '../widgets/profile_sections.dart';
import '../widgets/user_profile_box.dart';
import 'addresses/show_addresses.dart';
import 'coupons/empty_coupon.dart';
import 'notifications/empty_notification.dart';
import 'orders/empty_order.dart';
import 'wishlist/empty_wishlist.dart';

class ProfileScreen extends StatefulWidget {
    const ProfileScreen({super.key});
    // for route navigation
    static String id = 'ProfileScreen';

    @override
    State<ProfileScreen> createState() => _ProfileScreenState();
}

class _ProfileScreenState extends State<ProfileScreen> {
    // list of sections in the profile like: notifications, coupons,etc
    late ScrollController scrollController;

    final int _currentIndex = 4;

    int lastIndex = 4;

    @override
    void initState() {
        super.initState();
        scrollController = ScrollController();
    }

    @override
    void dispose() {
        scrollController.dispose();
        super.dispose();
    }

    @override
    Widget build(BuildContext context) {
        List<ProfileSections> profileSections = [
            ProfileSections(
                icon: Icons.notifications_none,

```

```

        text: 'Notifications',
        onTap: () => Navigator.pushNamed(context, EmptyNotification.id)),
    ProfileSections(
        icon: Icons.discount,
        text: 'Copouns',
        onTap: () => Navigator.pushNamed(context, EmptyCoupon.id)),
    ProfileSections(
        icon: Icons.receipt,
        text: 'My orders',
        onTap: () => Navigator.pushNamed(context, EmptyOrders.id)),
    ProfileSections(
        icon: Icons.favorite_border,
        text: 'Wishlist',
        onTap: () => Navigator.pushNamed(context, EmptyWishlist.id)),
    ProfileSections(
        icon: Icons.login_outlined,
        text: 'Logout',
        onTap: () {
            showDialogWidget(
                context,
                firstTitle: 'Are you sure',
                secondTitle: 'you want to logout?',
                firstTap: () => Navigator.pop(context),
                secondTap: () => Navigator.pushNamed(context, Login.id),
                firstButtonText: 'Cancel',
                secondButtonText: 'Sure',
            );
        }),
    ProfileSections(
        icon: Icons.person_remove_alt_1_outlined,
        text: 'Delete Account',
        onTap: () {
            showDialogWidget(context,
                firstTitle: 'Are you sure',
                secondTitle: 'you want to delete your account?',
                firstTap: () => Navigator.pop(context),
                secondTap: () => Navigator.pushNamed(context, Login.id),
                firstButtonText: 'Cancel',
                secondButtonText: 'Delete',
                firstTitleSize: 20,
                secondTitleSize: 20);
        }),
    ],
);
return Scaffold(
    appBar: AppBar(

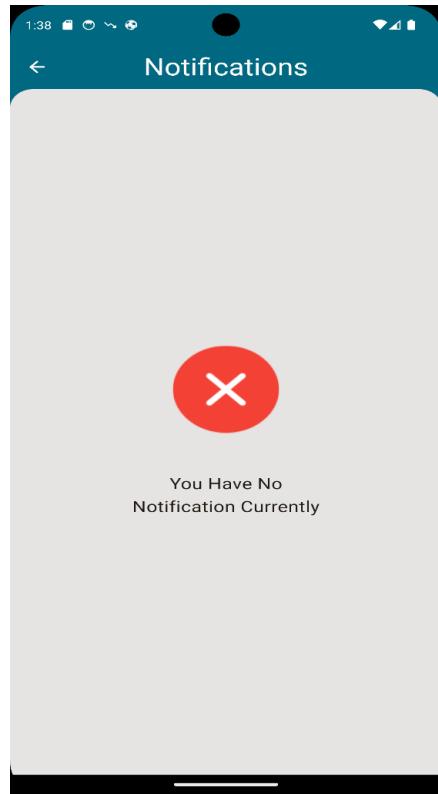
```

```

        //leadingIcon =====> is extracted widget to put icon in leading and
        custimize attributes
        leading: LeadingIcon(
            icon: Icons.home,
            onPressed: () => Navigator.pushNamed(context, Home.id),
            size: 35,
            horizontal: 15,
        ),
        title: const Text('profile'),
    ),
    body: Background(
        child: Column(
            children: [
                // gap is for make space
                const Gap(35),
                //its widget has profile picture, edit profile picture icon and name
of user
                const UserProfileBox(),
                const Gap(50),
                // listview of sections in the profile that build list by index
                SizedBox(
                    height: 500,
                    child: ListView.builder(
                        itemBuilder: (context, index) => profileSections[index],
                        itemCount: profileSections.length,
                        shrinkWrap: true,
                        //to cancel auto scroll
                        physics: const NeverScrollableScrollPhysics(),
                    ),
                ),
                ],
            ),
        ),
        bottomNavigationBar: CustomBottomNavBar(
            controller: scrollController,
            currentIndex: _currentIndex,
            lastIndex: lastIndex,
        ),
        extendBody: true,
    );
}
}

```

Notifications: show notifications for the user.



## Code Implementation:

```
import 'package:flutter/material.dart';
import 'package:gap/gap.dart';

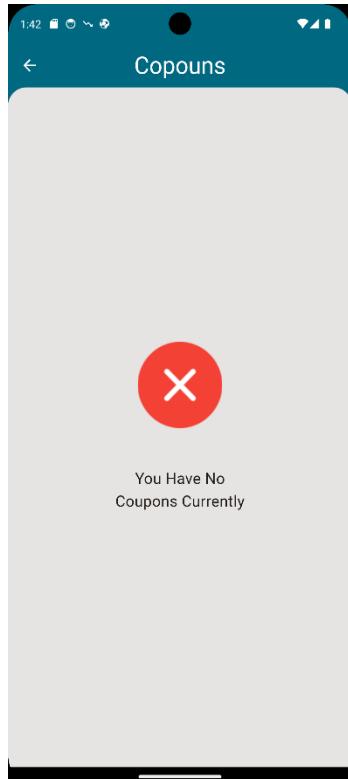
import '../../../../../utils/components/background.dart';
import '../../../../../utils/helper/constants/images.dart';

class EmptyNotification extends StatelessWidget {
  const EmptyNotification({super.key});
  static String id = 'empty_notification';
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Notifications'),
      ),
      body: const Background(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [

```

```
        Image(
            image: AssetImage(kEmptyState),
        ),
        Gap(50),
        Text(
            'You Have No',
            style: TextStyle(
                fontSize: 20,
            ),
        ),
        Text(
            'Notification Currently',
            style: TextStyle(
                fontSize: 20,
            ),
        ),
    ],
),
),
);
}
}
```

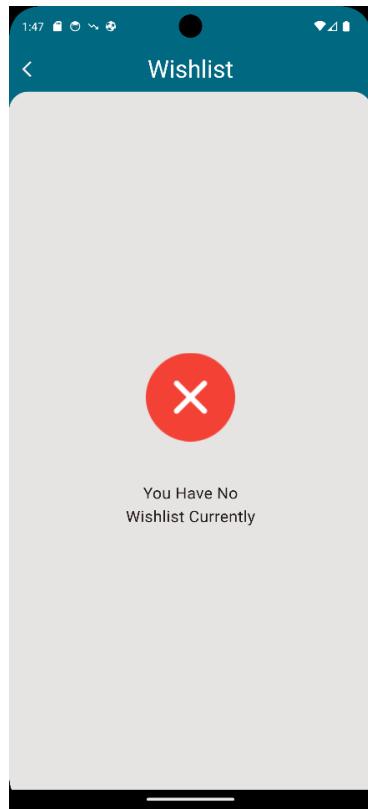
Coupons: show Coupons for the user •



```
import 'package:flutter/material.dart';
import 'package:gap/gap.dart';
import '../../../../../utils/components/background.dart';
import '../../../../../utils/helper/constants/colors.dart';
import '../../../../../utils/helper/constants/images.dart';

class EmptyCoupon extends StatelessWidget {
  const EmptyCoupon({super.key});
  static String id = 'EmptyCoupon';
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Copouns'),
      ),
      body: const Background(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Image(
              image: AssetImage(kEmptyState),
            ),
            Gap(50),
            Text(
              'You Have No',
              style: TextStyle(
                fontSize: 20,
              ),
            ),
            Text(
              'Coupons Currently',
              style: TextStyle(
                fontSize: 20,
              ),
            )
          ],
        ),
      );
  }
}
```

## Wishlist: show the Wishlist of user.



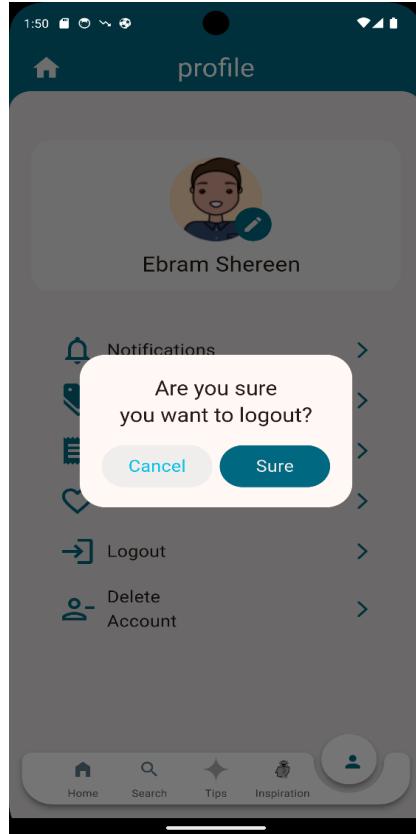
```
import 'package:flutter/material.dart';
import 'package:gap/gap.dart';

import '../../../../../utils/components/background.dart';
import '../../../../../utils/components/leading_icon.dart';
import '../../../../../utils/helper/constants/images.dart';

class EmptyWishlist extends StatelessWidget {
  const EmptyWishlist({super.key});
  static String id = 'EmptyWishlist';
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Wishlist'),
        leading: LeadingIcon(
          onPressed: () => Navigator.pop(context),
        ),
      ),
      body: const Background(
        child: Column(
          mainAxisAlignment: MainAxisAlignment
```

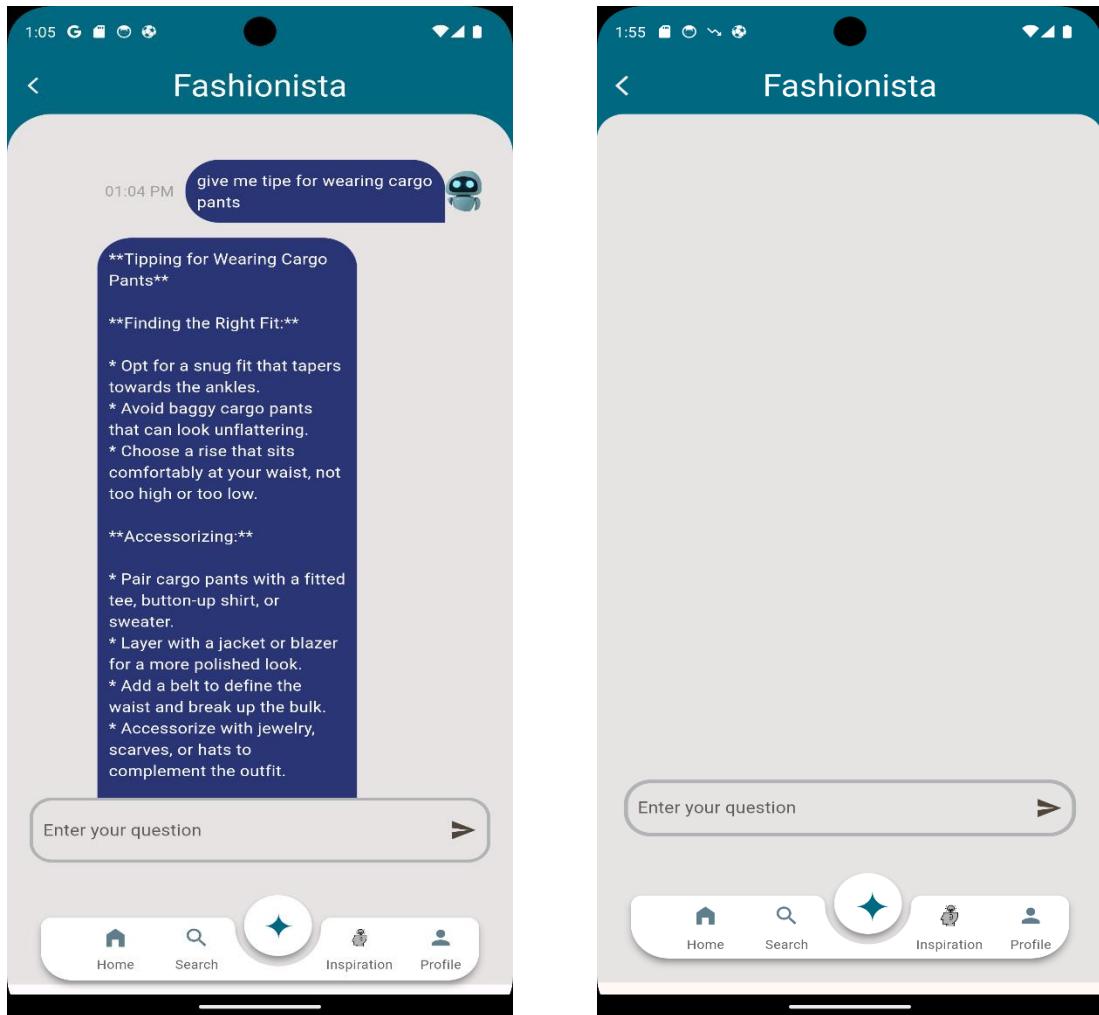
```
        children: [
          Image(
            image: AssetImage(kEmptyState),
          ),
          Gap(50),
          Text(
            'You Have No',
            style: TextStyle(
              fontSize: 20,
            ),
          ),
          Text(
            'Wishlist Currently',
            style: TextStyle(
              fontSize: 20,
            ),
          ),
        ],
      ),
    );
  }
}
```

Logout: show logout dialogue.



## 5.1.5 ChatBot

Chatbot: integrate with the Gemini API to help users.



## Code Implementation:

```
// chatbot_model.dart
class ChatBotModel {
    String message;
    String date;
    bool isChatBot;

    ChatBotModel({
        required this.message,
        required this.date,
        required this.isChatBot,
```

```

    });
}

// chatbot_service.dart
import 'package:google_generative_ai/google_generative_ai.dart';
import 'package:on_budget/src/utils/helper/constants/api.dart';
import 'package:on_budget/src/utils/helper/constants/chat_date.dart';
import '../models/chatbot_model.dart';

class ChatBotService {
  Future<ChatBotModel> sendMessage(String msg) async {
    final model = GenerativeModel(
      model: ChatbotApi().kChatBotVersion,
      apiKey: ChatbotApi().kChatBotApiKey,
    );
    final content = Content.text(msg);
    final response = await model.generateContent([content]);

    return ChatBotModel(
      message: response.text ?? '',
      date: ChatDate().date(),
      isChatBot: true,
    );
  }
}

// chatbot_repository.dart
import '../models/chatbot_model.dart';
import '../service/chatbot_service.dart';

class ChatBotRepository {
  final ChatBotService _chatBotService;

  ChatBotRepository(this._chatBotService);

  Future<ChatBotModel> sendMessage(String msg) async {
    return await _chatBotService.sendMessage(msg);
  }
}

import
'package:on_budget/src/modules/customer_modules/chatbot/data/models/chatbot_model
.dart';

class ChatBotState {

```

```

final List<ChatBotModel> messagesList;

ChatBotState(this.messagesList);
}

class ChatBotInitial extends ChatBotState {
    ChatBotInitial() : super([]);
}

class ChatBotLoading extends ChatBotState {
    ChatBotLoading(List<ChatBotModel> messagesList) : super(messagesList);
}

class ChatBotSuccess extends ChatBotState {
    ChatBotSuccess(List<ChatBotModel> messagesList) : super(messagesList);
}

class ChatBotError extends ChatBotState {
    final String error;

    ChatBotError(List<ChatBotModel> messagesList, this.error)
        : super(messagesList);
}

import 'package:bloc/bloc.dart';
import
'package:on_budget/src/modules/customer_modules/chatbot/logic/chatbot_states.dart'
';
import '../data/models/chatbot_model.dart';
import '../data/repository/chatbot_repository.dart';

class ChatBotCubit extends Cubit<ChatBotState> {
    final ChatBotRepository _chatBotRepository;

    ChatBotCubit(this._chatBotRepository) : super(ChatBotInitial());

    void sendMessage(String message, List<ChatBotModel> messagesList) async {
        try {
            emit(ChatBotLoading(messagesList));
            final response = await _chatBotRepository.sendMessage(message);
            messagesList.add(ChatBotModel(
                message: message,
                date: DateTime.now().toString(),
                isChatBot: false,
            ));
        }
    }
}

```

```

        messagesList.add(response);
        emit(ChatBotSuccess(messagesList));
    } catch (e) {
        emit(ChatBotError(messagesList, e.toString()));
    }
}

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';
import
'package:on_budget/src/modules/customer_modules/chatbot/data/models/chatbot_model
.dart';
import
'package:on_budget/src/modules/customer_modules/chatbot/data/repository/chatbot_r
epository.dart';
import
'package:on_budget/src/modules/customer_modules/chatbot/data/service/chatbot_serv
ice.dart';
import
'package:on_budget/src/modules/customer_modules/chatbot/logic/chatbot_cubit.dart'
;
import
'package:on_budget/src/modules/customer_modules/chatbot/logic/chatbot_states.dart'
;
import
'package:on_budget/src/modules/customer_modules/chatbot/presentation/widgets/chat
bot_message.dart';
import
'package:on_budget/src/modules/customer_modules/chatbot/presentation/widgets/mess
age_in_chat.dart';
import 'package:on_budget/src/utils/components/background.dart';
import 'package:on_budget/src/utils/components/bottom_bar/bottom_navber.dart';
import 'package:on_budget/src/utils/components/leading_icon.dart';
import 'package:on_budget/src/utils/helper/constants/colors.dart';
import ' ../../../../../../utils/components/text_field_widget.dart';

class ChatBotView extends StatelessWidget {
    const ChatBotView({Key? key}) : super(key: key);
    static String id = 'chatbot';

    @override
    Widget build(BuildContext context) {

```

```

final TextEditingController controller = TextEditingController();
final ScrollController scrollController = ScrollController();
final List<ChatBotModel> messagesList = [];

return Scaffold(
  appBar: AppBar(
    title: const Text('Fashionista'),
    leading: const LeadingIcon(),
  ),
  body: Background(
    child: BlocProvider(
      create: (context) =>
          ChatBotCubit(ChatBotRepository(ChatBotService())),
      child: Column(
        children: [
          Gap(ScreenUtil().setHeight(20)),
          Expanded(
            child: BlocBuilder<ChatBotCubit, ChatBotState>(
              builder: (context, state) {
                if (state is ChatBotLoading) {
                  return const Center(child: CircularProgressIndicator());
                }
                if (state is ChatBotError) {
                  return Center(child: Text('Error: ${state.error}'));
                }
                final messages = state.messagesList.reversed.toList();
                return ListView.builder(
                  reverse: true,
                  itemCount: messages.length,
                  itemBuilder: (context, index) {
                    final msg = messages[index];
                    return msg.isChatBot
                      ? ChatbotMessage(
                          message: msg.message,
                          isChatBot: msg.isChatBot,
                          date: msg.date,
                        )
                      : UserMessage(
                          message: msg.message,
                          isChatBot: msg.isChatBot,
                          date: msg.date,
                        );
                  },
                );
              },
            );
          ],
        ),
      ),
    ),
  ),
);

```

```

        ),
    ),
    TextFieldWidget(
        controller: controller,
        hintText: 'Enter your question',
        onSubmitted: (value) {
            if (value.isNotEmpty) {
                context
                    .read<ChatBotCubit>()
                    .sendMessage(value, messagesList);
                controller.clear();
            }
        },
        suffixIcon: IconButton(
            onPressed: () {
                final message = controller.text;
                if (message.isNotEmpty) {
                    context
                        .read<ChatBotCubit>()
                        .sendMessage(message, messagesList);
                    controller.clear();
                }
            },
            icon: const Icon(Icons.send),
        ),
    ),
],
),
),
),
),
),
),
bottomNavigationBar: Padding(
    padding: const EdgeInsets.only(bottom: 10).r,
    child: BottomAppBar(
        color: kThirdColor,
        elevation: 0,
        height: 100.h,
        child: CustomBottomNavBar(
            currentIndex: 2,
            controller: scrollController,
            lastIndex: 2,
        ),
    ),
),
);
}

```

```
}

class MyWidget extends StatelessWidget {
    const MyWidget({super.key});

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: SizedBox(
                width: 200.w,
                height: 300.h,
            ),
        );
    }
}

// chatbot_message.dart
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';
import 'package:on_budget/src/utils/helper/constants/api.dart';

class ChatbotMessage extends StatelessWidget {
    const ChatbotMessage({
        Key? key,
        required this.message,
        required this.isChatBot,
        required this.date,
    }) : super(key: key);

    final String message;
    final bool isChatBot;
    final String date;

    @override
    Widget build(BuildContext context) {
        return Row(mainAxisAlignment: MainAxisAlignment.start, children: [
            Padding(
                padding: const EdgeInsets.only(left: 10, top: 5, bottom: 10).r,
                child: Align(
                    child: Row(
                        children: [
                            CircleAvatar(
                                backgroundImage: AssetImage(ChatbotApi().kChatBot),
                                backgroundColor: Colors.transparent,
                            )
                        ],
                    ),
                ),
            )
        ]);
    }
}
```

```

),
Gap(ScreenUtil().setWidth(5)),
Container(
  padding: const EdgeInsets.all(10).w,
  constraints: BoxConstraints(maxWidth: 190.w),
  decoration: BoxDecoration(
    color: const Color(0xFF2a3575),
    borderRadius: BorderRadius.only(
      topLeft: const Radius.circular(30),
      topRight: const Radius.circular(30),
      bottomRight: Radius.circular(isChatBot ? 30 : 0),
      bottomLeft: Radius.circular(isChatBot ? 0 : 30),
    ).r,
  ),
  child: Text(
    message,
    style: TextStyle(color: Colors.white, fontSize: 12.sp),
  ),
),
const Gap(10),
],
),
),
),
),
Padding(
  padding: const EdgeInsets.only(right: 30).r,
  child: Text(
    date.toString(),
    style: const TextStyle(color: Colors.grey),
  ),
),
]);
}
}
// user_message.dart
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';
import 'package:on_budget/src/utils/helper/constants/api.dart';

class UserMessage extends StatelessWidget {
  const UserMessage({
    Key? key,
    required this.message,
    required this.isChatBot,
  }

```

```
        required this.date,
    }) : super(key: key);

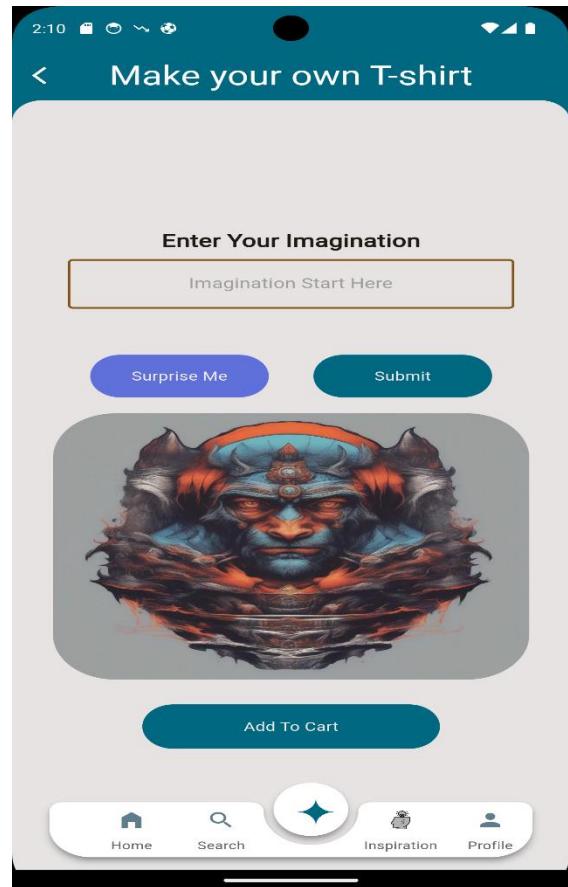
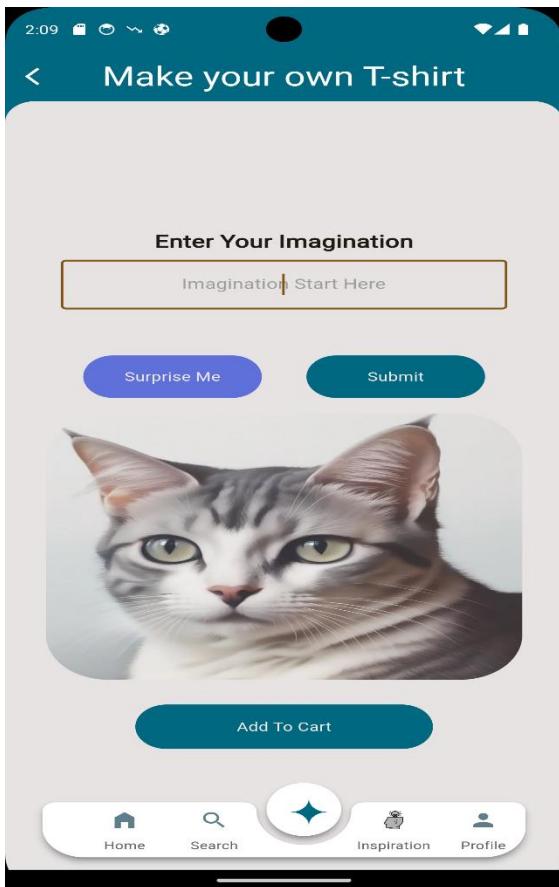
    final String message;
    final bool isChatBot;
    final String date;

    @override
    Widget build(BuildContext context) {
        return Row(mainAxisAlignment: MainAxisAlignment.end, children: [
            Padding(
                padding: const EdgeInsets.only(left: 10, top: 5, bottom: 10).r,
                child: Align(
                    child: Row(
                        children: [
                            Text(
                                date.toString(),
                                style: const TextStyle(color: Colors.grey),
                            ),
                            Gap(ScreenUtil().setWidth(10)),
                            Container(
                                padding: const EdgeInsets.all(10).w,
                                constraints: BoxConstraints(maxWidth: 190.w),
                                decoration: BoxDecoration(
                                    color: const Color(0xFF2a3575),
                                    borderRadius: BorderRadius.only(
                                        topLeft: const Radius.circular(30),
                                        topRight: const Radius.circular(30),
                                        bottomRight: Radius.circular(isChatBot ? 30 : 0),
                                        bottomLeft: Radius.circular(isChatBot ? 0 : 30),
                                    ).r,
                                ),
                                child: Text(
                                    message,
                                    style: TextStyle(color: Colors.white, fontSize: 15.sp),
                                ),
                            ),
                            Gap(ScreenUtil().setWidth(10)),
                            CircleAvatar(
                                backgroundImage: AssetImage(ChatbotApi().kChatBot),
                                backgroundColor: Colors.transparent,
                            ),
                        ],
                    ),
                ),
            ),
        ],
    ),
}
```

```
        ),
        Visibility(
            visible: isChatBot,
            child: Padding(
                padding: const EdgeInsets.only(right: 30),
                child: Text(
                    date.toString(),
                    style: const TextStyle(color: Colors.grey),
                ),
            ),
        ),
    ],
);
}
}
```

## 5.1.6 Generate Image

The model accept text in textField and get the image in results



## Code Implementation:

```
import 'dart:convert';
import 'dart:typed_data';

import 'package:http/http.dart' as http;
import 'package:http/http.dart';

class GenerateTshirtApi {
  Future<Uint8List> post(
    required String url,
    required Map<String, dynamic> body,
    required String token) async {
    Map<String, String> headers = {
      'Authorization': 'Bearer $token',
      "accept": "image/*"
    };
    Response response = await http.post(Uri.parse(url),
        body: jsonEncode(body), headers: headers);
    if (response.statusCode == 200) {
      return response.bodyBytes;
    } else {
      throw Exception('Failed to load image');
    }
  }
}
import 'dart:typed_data';

import 'package:card_loading/card_loading.dart';
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';
import 'package:on_budget/src/utils/helper/constants/api.dart';
import '../../../../../utils/components/bottom_bar/bottom_navber.dart';

import '../../../../../utils/components/background.dart';
import '../../../../../utils/components/button.dart';
import '../../../../../utils/helper/constants/colors.dart';
import '../../../../../data/service/generate_tshirt_api.dart';
import '../widgets/generate_t-shirt_result.dart';

class GenerateTshirt extends StatefulWidget {
  const GenerateTshirt({Key? key}) : super(key: key);

  static String id = 'GenerateT-shirt';
```

```

@Override
// ignore: library_private_types_in_public_api
_generateTshirtState createState() => _GenerateTshirtState();
}

class _GenerateTshirtState extends State<GenerateTshirt> {
  bool visible = false;
  String? custimizeImagination;
  final String defaultImagination = 'TshirtDesignAF, T Shirt Design';
  Future<Uint8List>? _imageFuture;
  String? errorMessage;
  ScrollController scrollController = ScrollController();

  final int _currentIndex = 3;

  int lastIndex = 3;

  @override
  void initState() {
    super.initState();
    _fetchImage(imagination: custimizeImagination ?? defaultImagination);
    scrollController = ScrollController();
  }

  @override
  void dispose() {
    scrollController.dispose();

    super.dispose();
  }

  void _fetchImage({required String imagination}) {
    setState(() {
      _imageFuture = GenerateTshirtApi().post(
        url: GenerateImages().kGenerateImageApi,
        token: GenerateImages().kGenerateImageToken,
        body: {
          'inputs': imagination,
          'timestamp': DateTime.now().millisecondsSinceEpoch * .0000007
        },
      );
    });
  }
}

```

```

@Override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Make your own T-shirt'),
      leading: IconButton(
        onPressed: () => Navigator.pop(context),
        icon: const Icon(Icons.arrow_back_ios),
        color: Colors.white,
      ),
    ),
    body: Background(
      child: SingleChildScrollView(
        child: Column(
          children: [
            SizedBox(height: 100.h),
            const Center(
              child: Text(
                'Enter Your Imagination',
                style: TextStyle(
                  fontSize: 20,
                  fontWeight: FontWeight.w700,
                ),
            ),
            ),
            SizedBox(height: 5.h),
            SizedBox(
              width: 300.w,
              height: 90.h,
              child: TextField(
                onChanged: (value) {
                  setState(() {
                    custimizeImagination = value;
                  });
                },
                enableSuggestions: true,
                textAlign: TextAlign.center,
                decoration: const InputDecoration(
                  border: OutlineInputBorder(),
                  hintText: 'Imagination Start Here',
                  hintStyle: TextStyle(color: Colors.grey),
                ),
            ),
          ],
        ),
      ),
    ),
  );
}

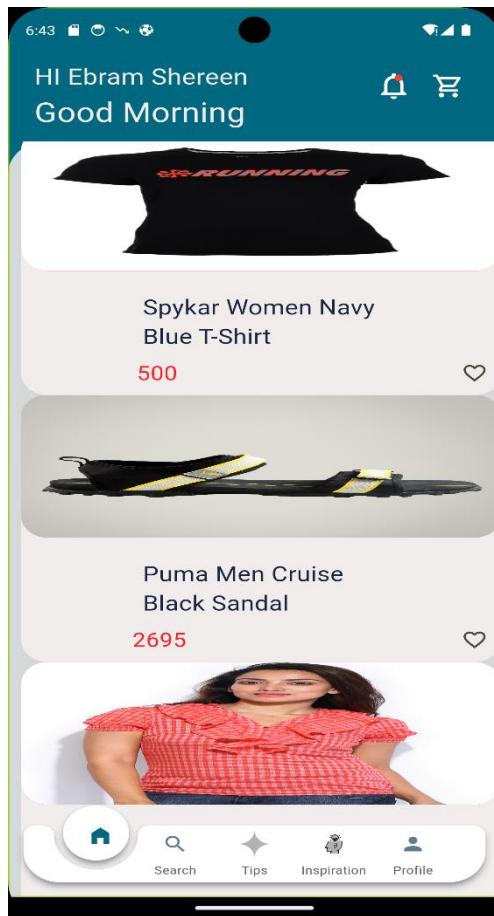
```

```
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Button(
        textSize: 14,
        text: 'Surprise Me',
        tap: () {
          setState(() {
            _fetchImage(imagination: defaultImagination);
            visible = true;
          });
        },
        width: 120.w,
        colorBtn: const Color(0xCC3D54DA),
        colorTxt: kSecondaryColor,
        height: 40.h,
      ),
      SizedBox(width: 30.w),
      Button(
        textSize: 14,
        text: 'Submit',
        tap: () {
          if (custimizeImagination != null) {
            _fetchImage(imagination: custimizeImagination!);
            visible = true;
          }
        },
        width: 120.w,
        colorBtn: kPrimaryColor,
        colorTxt: kSecondaryColor,
        height: 40.h,
      ),
    ],
  ),
  const Gap(10),
  Visibility(
    visible: visible,
    child: FutureBuilder<Uint8List>(
      future: _imageFuture,
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return CardLoading(
            height: 300,
            width: 320.w,
            borderRadius: BorderRadius.circular(50),
          );
        }
      }
    )
  )
}
```



```
Uint8List bytes;
@Override
Widget build(BuildContext context) {
    return Column(
        children: [
            Padding(
                padding: const EdgeInsets.all(8.0),
                child: Container(
                    decoration: const BoxDecoration(
                        borderRadius: BorderRadius.all(Radius.circular(50)),
                    ),
                    width: 320.w,
                    height: 300,
                    clipBehavior: Clip.antiAlias,
                    child: Image.memory(
                        bytes,
                        fit: BoxFit.fill,
                    ),
                ),
            ),
            Padding(
                padding: const EdgeInsets.all(20),
                child: Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                        Button(
                            textSize: 14,
                            text: 'Add To Cart',
                            tap: () {},
                            width: 200.w,
                            colorBtn: kPrimaryColor,
                            colorTxt: kSecondaryColor,
                            height: 50,
                        ),
                    ],
                ),
            ],
        ],
    );
}
```

## 5.1.7 Home



## Code Implementation:

```
class Products {  
    int? id;  
    String? productName;  
    String? productDescription;  
    int? unitPrice;  
    String? color;  
    String? categoryName;  
    String? supplierHandle;  
    List<Pictures>? pictures;  
  
    Products(  
        {this.id,  
        this.productName,  
        this.productDescription,  
        this.unitPrice,  
        this.color,  
        this.categoryName,  
        this.supplierHandle,  
        this.pictures});  
}
```

```

        this.color,
        this.categoryName,
        this.supplierHandle,
        this.pictures));

Products.fromJson(Map<String, dynamic> json) {
    id = json['id'];
    productName = json['productName'];
    productDescription = json['productDescription'];
    unitPrice = json['unitPrice'];
    color = json['color'];
    categoryName = json['categoryName'];
    supplierHandle = json['supplierHandle'];
    if (json['pictures'] != null) {
        pictures = <Pictures>[];
        json['pictures'].forEach((v) {
            pictures!.add(Pictures.fromJson(v));
        });
    }
}

class Pictures {
    String? front;
    String? back;

    Pictures({this.front, this.back});

    Pictures.fromJson(Map<String, dynamic> json) {
        front = json['front'];
        back = json['back'];
    }
}

import 'dart:convert';

import 'package:http/http.dart';
import 'package:http/http.dart' as http;
import 'package:on_budget/src/utils/helper/constants/api.dart';

class WatchesCategoryService {
    Future<Map<String, dynamic>> getTShirtCategoryService() async {
        Response response = await http.get(
            Uri.parse(HomeApi().getWatchesCategoryProducts),
            headers: {'Accept': 'text/plain'},

```

```

    );
    print(response.body);
    return json.decode(response.body);
}
}

import
'package:on_budget/src/modules/customer_modules/home/data/models/products_by_cate
gory.dart';
import
'package:on_budget/src/modules/customer_modules/home/data/service/home_categories
/watches_category_service.dart';

class WatchesCategoryRepository {
    WatchesCategoryService watchesCategoryService;
    WatchesCategoryRepository({
        required this.watchesCategoryService,
    });
    Future<Products> getWatchesCategory() async {
        final data = await watchesCategoryService.getTShirtCategoryService();
        return Products.fromJson(data);
    }
}

// ignore_for_file: public_member_api_docs, sort_constructors_first
import
'package:on_budget/src/modules/customer_modules/home/data/models/products_by_cate
gory.dart';

class WatchesStates {}

class WatchesInitial extends WatchesStates {}

class WatchesWaiting extends WatchesStates {}

class WatchesLoaded extends WatchesStates {
    Products productsModel;
    WatchesLoaded({
        required this.productsModel,
    });
}

class WatchesFailure extends WatchesStates {
    String error;
    WatchesFailure({

```

```

        required this.error,
    });
}

import 'package:flutter_bloc/flutter_bloc.dart';
import
'package:on_budget/src/modules/customer_modules/home/data/repository/home_categor
ies/watches_category_repository.dart';
import
'package:on_budget/src/modules/customer_modules/home/logic/cubit/watches_category
_cubit/watches_states.dart';

class WatchesCubit extends Cubit<WatchesStates> {
    WatchesCategoryRepository watchesCategoryRepository;
    WatchesCubit({required this.watchesCategoryRepository})
        : super(WatchesInitial());

    Future<void> getWatchesCubit() async {
        emit(WatchesWaiting());
        try {
            final products = await watchesCategoryRepository.getWatchesCategory();
            emit(WatchesLoaded(productsModel: products));
        } on Exception catch (e) {
            emit(WatchesFailure(
                error: 'There was an error in watches cubit and the error is: $e'));
        }
    }
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';
import 'package:http/http.dart' as http;
import
'package:on_budget/src/modules/customer_modules/home/presentation/widgets/watches
/watches_listview.dart';
import 'dart:convert';
import '../../../../../utils/components/bottom_bar/bottom_navber.dart';
import '../../../../../widgets/home_sections/pants.dart';
import '../../../../../widgets/home_sections/recommended_items_listview.dart';
import '../../../../../widgets/home_sections/see_all.dart';
import '../../../../../widgets/home_sections/shoes_listview.dart';
import '../../../../../widgets/home_sections/underwear_listview.dart';
import '../../../../../utils/components/home_background.dart';

```

```
class Home extends StatefulWidget {
  const Home({Key? key}) : super(key: key);

  static String id = 'Home';

  @override
  State<Home> createState() => _HomeState();
}

class _HomeState extends State<Home> {
  ScrollController scrollController = ScrollController();
  final int _currentIndex = 0;
  int lastIndex = 0;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: HomeBackground(
        child: SingleChildScrollView(
          physics: const BouncingScrollPhysics(),
          controller: scrollController,
          child: Column(
            children: [
              Gap(ScreenUtil().setHeight(40)),
              const SeeAll(text: 'Recommended Items'),
              Gap(ScreenUtil().setHeight(10)),
              const RecommendedItemsListView(),
              Gap(ScreenUtil().setHeight(10)),
              const SeeAll(text: 'T-shirts'),
              Gap(ScreenUtil().setHeight(10)),
              const WatchesItemsListView(),
              Gap(ScreenUtil().setHeight(10)),
              const SeeAll(text: 'Pants'),
              Gap(ScreenUtil().setHeight(10)),
              const PantsItemsListView(),
              Gap(ScreenUtil().setHeight(10)),
              const SeeAll(text: 'Shoes'),
              Gap(ScreenUtil().setHeight(10)),
              const ShoesItemsListView(),
              Gap(ScreenUtil().setHeight(10)),
              const SeeAll(text: 'Underwear'),
              Gap(ScreenUtil().setHeight(10)),
              const UnderwearItemsListView(),
              Gap(ScreenUtil().setHeight(10)),
              const SeeAll(text: 'Blazer'),
            ],
          ),
        ),
      ),
    );
  }
}
```

```

        Gap(ScreenUtil().setHeight(10)),
        const UnderwearItemsListView(),
        const SizedBox(height: 40)
    ],
),
),
),
bottomNavigationBar: CustomBottomNavBar(
    currentIndex: _currentIndex,
    lastIndex: lastIndex,
    controller: scrollController,
),
extendBody: true,
);
}
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

import 'home_items.dart';

class BlazerItemsListView extends StatelessWidget {
const BlazerItemsListView({
    super.key,
});
@Override
Widget build(BuildContext context) {
    return SizedBox(
        height: 220.h,
        child: ListView.builder(
            itemBuilder: (context, index) => HomeItems(),
            physics: const BouncingScrollPhysics(),
            scrollDirection: Axis.horizontal,
        ),
    );
}
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';
import
'package:on_budget/src/modules/customer_modules/home/data/models/products_by_cate
gory.dart';
import '../../../../../utils/helper/constants/colors.dart';

```

```
import '../../../../../utils/helper/functions/responsive.dart';

class HomeItems extends StatefulWidget {
  const HomeItems({
    Key? key,
  }) : super(key: key);

  @override
  State<HomeItems> createState() => _HomeItemsState();
}

class _HomeItemsState extends State<HomeItems> {
  bool favorite = false;

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.only(left: 20).r,
      child: Container(
        width: 180.w,
        decoration: BoxDecoration(
          color: kSecondaryColor,
          borderRadius: const BorderRadius.all(Radius.circular(20)).w,
        ),
        child: Column(
          children: [
            Container(
              width: 180.w,
              height: 130.h,
              decoration: BoxDecoration(
                borderRadius: const BorderRadius.all(Radius.circular(20)).w,
                color: Colors.red,
                image: const DecorationImage(
                  fit: BoxFit.contain,
                  image: NetworkImage(
                    'https://avatars.githubusercontent.com/u/95384301?v=4'),
                ),
            ),
          ],
        ),
        Gap(ScreenUtil().setHeight(20)),
        Text(
          'widget',
          style: TextStyle(
            color: const Color(0xFF1C2340),
            fontSize: 19.sp,
```

```
        fontWeight: FontWeight.w500,
    ),
),
Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
        Gap(ScreenUtil().setHeight(5)),
        Text(
            '\$20',
            style: TextStyle(
                color: kSalePrice,
                fontSize: 18.sp,
                fontWeight: FontWeight.w400,
            ),
        ),
        Gap(ScreenUtil().setWidth(40)),
        favorite
            ? IconButton(
                enableFeedback: false,
                icon: favorite
                    ? const Icon(
                        color: Colors.red,
                        Icons.favorite,
                    )
                    : const Icon(Icons.favorite_border),
                highlightColor: Colors.transparent,
                onPressed: () {
                    setState(() {
                        favorite = !favorite;
                    });
                },
            )
            : IconButton(
                enableFeedback: false,
                icon: favorite
                    ? const Icon(
                        color: Colors.red,
                        Icons.favorite,
                    )
                    : const Icon(Icons.favorite_border),
                highlightColor: Colors.transparent,
                onPressed: () {
                    setState(() {
                        favorite = !favorite;
                    });
                },
            );
    ],
);
```

```

        },
    ),
],
),
),
),
);
}
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'home_items.dart';

class PantsItemsListView extends StatelessWidget {
  const PantsItemsListView({super.key});

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: 220.h,
      child: ListView.builder(
        itemCount: 5,
        itemBuilder: (context, index) => const HomeItems(),
        physics: const BouncingScrollPhysics(),
        scrollDirection: Axis.horizontal,
      ),
    );
  }
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

import 'home_items.dart';

class RecommendedItemsListView extends StatelessWidget {
  const RecommendedItemsListView({
    super.key,
  });

  @override
  Widget build(BuildContext context) {
    return SizedBox(

```

```

        height: 220.h,
        child: ListView.builder(
            shrinkWrap: true,
            itemBuilder: (context, index) => const HomeItems(),
            scrollDirection: Axis.horizontal,
            physics: const BouncingScrollPhysics(),
        ),
    );
}
}

import 'package:flutter/material.dart';
import 'package:flutter/widgets.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';

class SeeAll extends StatelessWidget {
    const SeeAll({
        Key? key,
        required this.text,
    }) : super(key: key);

    final String text;

    @override
    Widget build(BuildContext context) {
        return Padding(
            padding: const EdgeInsets.symmetric(horizontal: 20).w,
            child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                    SizedBox(
                        width: 200.w,
                        height: 20.h,
                        child: Text(
                            text,
                            style: TextStyle(
                                color: const Color(0xFF1C2340),
                                fontSize: 16.sp,
                                fontWeight: FontWeight.w700,
                            ),
                    ),
                ],
            ),
            GestureDetector(
                onTap: () {},

```

```

        child: Text(
            'See All +',
            style: TextStyle(
                color: const Color(0xFF8A8D9F),
                fontSize: 16.sp,
                fontWeight: FontWeight.w700,
            ),
        ),
    ),
),
),
],
),
);
}
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import
'package:on_budget/src/modules/customer_modules/home/presentation/widgets/home_se
ctions/home_items.dart';

class ShoesItemsListView extends StatelessWidget {
    const ShoesItemsListView({
        super.key,
    });
    @override
    Widget build(BuildContext context) {
        return SizedBox(
            height: 220.h,
            child: ListView.builder(
                itemBuilder: (context, index) => HomeItems(),
                physics: const BouncingScrollPhysics(),
                scrollDirection: Axis.horizontal,
            ),
        );
    }
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

import 'home_items.dart';

class TshirtItemsListView extends StatelessWidget {

```

```

const TshirtItemsListView({
    super.key,
});
@Override
Widget build(BuildContext context) {
    return SizedBox(
        height: 220.h,
        child: ListView.builder(
            itemBuilder: (context, index) => HomeItems(),
            physics: const BouncingScrollPhysics(),
            scrollDirection: Axis.horizontal,
        ),
    );
}
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';

import 'home_items.dart';

class UnderwearItemsListView extends StatelessWidget {
    const UnderwearItemsListView({
        super.key,
    });
    @override
    Widget build(BuildContext context) {
        return SizedBox(
            height: 220.h,
            child: ListView.builder(
                itemBuilder: (context, index) => const HomeItems(),
                physics: const BouncingScrollPhysics(),
                scrollDirection: Axis.horizontal,
            ),
        );
    }
}

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import
'package:on_budget/src/modules/customer_modules/home/data/models/products_by_category.dart';

```

```

import
'package:on_budget/src/modules/customer_modules/home/data/repository/home_categories/watches_category_repository.dart';
import
'package:on_budget/src/modules/customer_modules/home/data/service/home_categories/watches_category_service.dart';
import
'package:on_budget/src/modules/customer_modules/home/logic/cubit/watches_category_cubit/watches_cubit.dart';
import
'package:on_budget/src/modules/customer_modules/home/logic/cubit/watches_category_cubit/watches_states.dart';
import
'package:on_budget/src/modules/customer_modules/home/presentation/widgets/home_sections/home_items.dart';
import
'package:on_budget/src/modules/customer_modules/home/presentation/widgets/watches/watches_product.dart';

class WatchesItemsListView extends StatelessWidget {
  const WatchesItemsListView({
    super.key,
  });

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: 220.h,
      child: BlocProvider(
        create: (context) => WatchesCubit(
          watchesCategoryRepository: WatchesCategoryRepository(
            watchesCategoryService: WatchesCategoryService())),
        ..getWatchesCubit(),
        child: BlocBuilder<WatchesCubit, WatchesStates>(
          builder: (context, state) {
            if (state is WatchesWaiting) {
              return const Center(
                child: CircularProgressIndicator(),
              );
            } else if (state is WatchesLoaded) {
              return ListView.builder(
                itemCount: 5,
                itemBuilder: (context, index) => WatchesProduct(
                  productsModel: state.productsModel,
                ),
            );
          }
        ),
      ),
    );
  }
}

```

```

        physics: const BouncingScrollPhysics(),
        scrollDirection: Axis.horizontal,
    );
} else if (state is WatchesFailure) {
    return Center(
        child: Text(state.error),
    );
} else {
    return const Center(
        child: Text('Error'),
    );
}
},
),
),
);
}
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';
import
'package:on_budget/src/modules/customer_modules/home/data/models/products_by_cate
gory.dart';
import 'package:on_budget/src/utils/helper/constants/colors.dart';

class WatchesProduct extends StatefulWidget {
    WatchesProduct({Key? key, required this.productsModel}) : super(key: key);
    Products productsModel;

    @override
    State<WatchesProduct> createState() => _WatchesProductState();
}

class _WatchesProductState extends State<WatchesProduct> {
    bool favorite = false;

    @override
    Widget build(BuildContext context) {
        return Padding(
            padding: const EdgeInsets.only(left: 20).r,
            child: Container(
                width: 180.w,
                decoration: BoxDecoration(

```

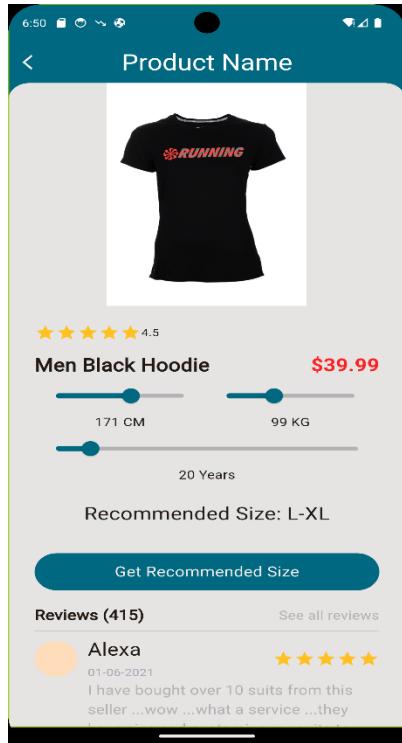
```

        color: kSecondaryColor,
        borderRadius: const BorderRadius.all(Radius.circular(20)).w,
    ),
    child: Column(
        children: [
            Container(
                width: 180.w,
                height: 130.h,
                decoration: BoxDecoration(
                    borderRadius: const BorderRadius.all(Radius.circular(20)).w,
                    color: Colors.red,
                    image: DecorationImage(
                        fit: BoxFit.contain,
                        image: NetworkImage(
                            widget.productsModel.pictures?.first.front ?? ''),
                    ),
                ),
            ),
            Gap(ScreenUtil().setHeight(20)),
            Text(
                widget.productsModel.productName ?? '',
                style: TextStyle(
                    color: const Color(0xFF1C2340),
                    fontSize: 19.sp,
                    fontWeight: FontWeight.w500,
                ),
            ),
            Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                    Gap(ScreenUtil().setHeight(5)),
                    Text(
                        '\$\${widget.productsModel.unitPrice ?? ''}',
                        style: TextStyle(
                            color: kSalePrice,
                            fontSize: 18.sp,
                            fontWeight: FontWeight.w400,
                        ),
                    ),
                    Gap(ScreenUtil().setWidth(40)),
                    favorite
                        ? IconButton(
                            enableFeedback: false,
                            icon: favorite
                                ? const Icon(

```

```
        color: Colors.red,
        Icons.favorite,
    )
: const Icon(Icons.favorite_border),
highlightColor: Colors.transparent,
onPressed: () {
    setState(() {
        favorite = !favorite;
    });
},
)
: IconButton(
    enableFeedback: false,
    icon: favorite
    ? const Icon(
        color: Colors.red,
        Icons.favorite,
    )
    : const Icon(Icons.favorite_border),
highlightColor: Colors.transparent,
onPressed: () {
    setState(() {
        favorite = !favorite;
    });
},
),
],
),
],
),
),
);
}
}
```

## 5.1.8 Product Details



## Code Implementation:

```
class SizePredictorModel {  
    final String? predictedSize;  
  
    SizePredictorModel({required this.predictedSize});  
    factory SizePredictorModel.fromJson(Map<String, String> response) {  
        return SizePredictorModel(predictedSize: response['predicted_size']);  
    }  
}  
  
import  
'package:on_budget/src/modules/customer_modules/home/data/models/size_predictor_m  
odel.dart';  
import  
'package:on_budget/src/modules/customer_modules/home/data/service/size_predictor/  
size_predictor_service.dart';  
import 'package:on_budget/src/utils/helper/constants/api.dart';  
  
class SizePredictorRepository {  
    SizePredictorService sizePredictorService;  
    SizePredictorRepository({
```

```

        required this.sizePredictorService,
    });

Future<SizePredictorModel> getSizePredict(
    {required Map<String, String> body}) async {
    final response = await sizePredictorService.getSizePredict(
        body: body, url: SizePredictor().url);
    return SizePredictorModel.fromJson(response);
}

import 'dart:convert';

import 'package:http/http.dart' as http;
import 'package:http/http.dart';

class SizePredictorService {
    Future<Map<String, String>> getSizePredict(
        {required Map<String, String> body, required String url}) async {
        Response response = await http.post(
            Uri.parse(url),
            body: jsonEncode(body),
            headers: {'Content-Type': 'application/x-www-form-urlencoded'},
        );
        try {
            return json.decode(response.body);
        } on Exception catch (e) {
            throw Exception('There was an error in Size predict service. $e');
        }
    }
}

import 'package:flutter_bloc/flutter_bloc.dart';
import
'package:on_budget/src/modules/customer_modules/home/data/repository/size_predictor/size_predictor_repository.dart';
import
'package:on_budget/src/modules/customer_modules/home/logic/cubit/size_predictor/size_predictor_states.dart';

class SizePredictorCubit extends Cubit<SizePredictorStates> {
    SizePredictorRepository sizePredictorRepository;
    SizePredictorCubit({required this.sizePredictorRepository})
        : super(SizePredictorInitial());
}

```

```

Future<void> getSizePredictor(double height, double weight, double age) async {
  emit(SizePredictorWaiting());
  try {
    final result = await sizePredictorRepository.getSizePredict(
      body: {
        'height': height.toString(),
        'weight': weight.toString(),
        'age': age.toString(),
      },
    );
    emit(SizePredictorLoaded(sizePredictorModel: result));
  } on Exception catch (e) {
    emit(SizePredictorError('There was an error in SizePredictorCubit and the
error is $e'));
  }
}
}

// ignore_for_file: public_member_api_docs, sort_constructors_first
import
'package:on_budget/src/modules/customer_modules/home/data/models/size_predictor_m
odel.dart';

class SizePredictorStates {}

class SizePredictorInitial extends SizePredictorStates {}

class SizePredictorWaiting extends SizePredictorStates {}

class SizePredictorLoaded extends SizePredictorStates {
  SizePredictorModel sizePredictorModel;
  SizePredictorLoaded({
    required this.sizePredictorModel,
  });
}

class SizePredictorError extends SizePredictorStates {
  final String message;
  SizePredictorError(this.message);
}

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';

```

```

import
'package:on_budget/src/modules/customer_modules/home/data/repository/size_predictor/size_predictor_repository.dart';
import
'package:on_budget/src/modules/customer_modules/home/data/service/size_predictor/size_predictor_service.dart';
import
'package:on_budget/src/modules/customer_modules/home/logic/cubit/size_predictor/size_predictor_cubit.dart';
import
'package:on_budget/src/modules/customer_modules/home/logic/cubit/size_predictor/size_predictor_states.dart';
import
'package:on_budget/src/modules/customer_modules/home/presentation/widgets/product_details/stars/five_stars.dart';
import 'package:on_budget/src/utils/components/background.dart';
import 'package:on_budget/src/utils/components/button.dart';
import 'package:on_budget/src/utils/components/leading_icon.dart';
import 'package:on_budget/src/utils/helper/constants/colors.dart';
import
'package:on_budget/src/modules/customer_modules/home/presentation/widgets/product_details/sliders/size_sliders.dart';

class ProductsDetails extends StatefulWidget {
  const ProductsDetails({super.key});
  static String id = 'productsDetails';

  @override
  State<ProductsDetails> createState() => _ProductsDetailsState();
}

class _ProductsDetailsState extends State<ProductsDetails> {
  double age = 10, weight = 55, height = 140;

  @override
  Widget build(BuildContext context) {
    return BlocProvider(
      create: (context) => SizePredictorCubit(
        sizePredictorRepository: SizePredictorRepository(
          sizePredictorService: SizePredictorService(),
        ),
      ),
      child: Scaffold(
        appBar: AppBar(
          title: const Text('Product Name'),

```

```
        leading: LeadingIcon(
            onPressed: () => Navigator.pop(context),
        ),
    ),
    body: Background(
        horizontalPadding: 0,
        verticalPadding: 0,
        child: SingleChildScrollView(
            child: Column(
                children: [
                    Container(
                        width: double.infinity,
                        height: 300,
                        decoration: BoxDecoration(
                            image: const DecorationImage(
                                image: AssetImage('lib/assets/10026.jpg'))),
                        borderRadius: const BorderRadius.only(
                            topLeft: Radius.circular(20),
                            topRight: Radius.circular(20),
                        ).r,
                    ),
                ],
            ),
            Padding(
                padding: const EdgeInsets.symmetric(horizontal: 30),
                child: Column(
                    children: [
                        Gap(ScreenUtil().setWidth(20)),
                        const Row(
                            children: [
                                FiveStars(),
                                Text('4.5'),
                            ],
                        ),
                        const Gap(15),
                        Row(
                            children: [
                                Text(
                                    'Men Black Hoodie',
                                    style: TextStyle(
                                        fontSize: 20.sp,
                                        fontWeight: FontWeight.w700,
                                    ),
                                ),
                                const Spacer(),
                                Text(

```

```

        '\$39.99',
        style: TextStyle(
            color: const Color(0xFFFFE2121),
            fontSize: 20.sp,
            fontFamily: 'Roboto',
            fontWeight: FontWeight.w700,
        ),
    ),
),
],
),
),
SizeSliders(
    valueHeight: height,
    valueWeight: weight,
    valueAge: age,
    onChanged: (newHeight, newWeight, newAge) {
        setState(() {
            height = newHeight;
            weight = newWeight;
            age = newAge;
        });
    },
),
Gap(ScreenUtil().setHeight(20)),
BlocProvider(
    create: (context) => SizePredictorCubit(
        sizePredictorRepository: SizePredictorRepository(
            sizePredictorService: SizePredictorService())),
    child: BlocBuilder<SizePredictorCubit,
        SizePredictorStates>(
            builder: (context, state) {
                if (state is SizePredictorLoaded) {
                    return Text(
                        'Recommended Size:
${state.sizePredictorModel.predictedSize}',
                        style: const TextStyle(fontSize: 24),
                    );
                } else if (state is SizePredictorWaiting) {
                    return const CircularProgressIndicator();
                } else {
                    return const Text(
                        'Recommended Size: L-XL',
                        style: TextStyle(fontSize: 24),
                    );
                }
            },
        ),
),

```



```

        const Spacer(),
        Text(
            'See all reviews',
            style: TextStyle(
                color: kHintColor,
                fontSize: 14.sp,
                fontWeight: FontWeight.w400,
            ),
        ),
    ],
),
const Divider(),
Row(
    children: [
        CircleAvatar(
            radius: ScreenUtil().radius(20),
        ),
        Gap(ScreenUtil().setWidth(10)),
        Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
                Text(
                    'Alexa',
                    style: TextStyle(
                        fontSize: 20.sp,
                        fontWeight: FontWeight.w400,
                    ),
                ),
                Gap(ScreenUtil().setHeight(3)),
                Text(
                    '01-06-2021',
                    style: TextStyle(
                        color: const Color(0xFFABABB7),
                        fontSize: 12.sp,
                        fontWeight: FontWeight.w400,
                    ),
                ),
            ],
        ),
        const Spacer(),
        const FiveStars(),
    ],
),
Row(
    mainAxisAlignment: MainAxisAlignment.center,

```

```

        children: [
            Gap(ScreenUtil().setWidth(30)),
            SizedBox(
                width: 255.w,
                height: 100.h,
                child: Text(
                    'I have bought over 10 suits from this seller ...wow ...what a
service ...they have size and customize my suits to a perfect and',
                    style: TextStyle(
                        color: kHintColor,
                        fontSize: 15.sp,
                        fontWeight: FontWeight.w400,
                    ),
                ),
            ),
        ],
    ],
),
],
);
}
}

import 'package:flutter/material.dart';

import '../../../../../utils/helper/constants/colors.dart';

class ProductSlider extends StatefulWidget {
ProductSlider({
    super.key,
    required this.value,
    required this.min,
    required this.max,
    required this.division,
    required this.unit,
    required this.onChanged,
});
double value;
final double min;
final double max;
final int division;
final String unit;
void Function(double)? onChanged;
@Override
State<ProductSlider> createState() => _ProductSliderState();
}

```

```

class _ProductSliderState extends State<ProductSlider> {
  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        SliderTheme(
          data: SliderThemeData(
            trackHeight: 5,
            tickMarkShape: SliderTickMarkShape.noTickMark,
            activeTrackColor: kPrimaryColor,
            inactiveTrackColor: kHintColor,
            valueIndicatorColor: kPrimaryColor,
            thumbColor: const Color(0xFF006880)),
        child: Slider(
          value: widget.value,
          onChanged: widget.onChanged,
          min: widget.min,
          max: widget.max,
          mouseCursor: MouseCursor.uncontrolled,
          divisions: widget.division,
        ),
      ),
      Text(
        '${widget.value.toInt()} ${widget.unit}',
        style: const TextStyle(
          fontSize: 16,
          fontWeight: FontWeight.w400,
        ),
      )
    ],
  );
}

import 'package:flutter/material.dart';
import
'package:on_budget/src/modules/customer_modules/home/presentation/widgets/product
_details/sliders/product_slider.dart';

class SizeSliders extends StatefulWidget {
  final double valueHeight;
  final double valueWeight;
  final double valueAge;
  final Function(double, double, double) onChanged;
}

```

```

SizeSliders({
    required this.valueHeight,
    required this.valueWeight,
    required this.valueAge,
    required this.onChanged,
});

@Override
State<SizeSliders> createState() => _SizeSlidersState();
}

class _SizeSlidersState extends State<SizeSliders> {
    late double height;
    late double weight;
    late double age;

    @override
    void initState() {
        super.initState();
        height = widget.valueHeight;
        weight = widget.valueWeight;
        age = widget.valueAge;
    }

    @override
    Widget build(BuildContext context) {
        return Column(
            children: [
                Row(
                    children: [
                        ProductSlider(
                            value: height,
                            min: 130,
                            max: 200,
                            division: 70,
                            unit: 'CM',
                            onChanged: (value) {
                                setState(() {
                                    height = value;
                                });
                                widget.onChanged(height, weight, age);
                            },
                        ),
                        ProductSlider(

```

```

        value: weight,
        min: 40,
        max: 200,
        division: 70,
        unit: 'KG',
        onChanged: (value) {
            setState(() {
                weight = value;
            });
            widget.onChanged(height, weight, age);
        },
    ],
),
),
ProductSlider(
    value: age,
    min: 10,
    max: 100,
    division: 70,
    unit: 'Years',
    onChanged: (value) {
        setState(() {
            age = value;
        });
        widget.onChanged(height, weight, age);
    },
),
],
);
}
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';
import
'package:on_budget/src/modules/customer_modules/home/presentation/widgets/product
_details/stars/five_stars.dart';
import 'package:on_budget/src/utils/helper/constants/colors.dart';

class ProductsReviews extends StatelessWidget {
const ProductsReviews({super.key});

@Override
Widget build(BuildContext context) {

```

```

return Column(
  children: [
    Row(
      children: [
        Text(
          'Reviews (415)',
          style: TextStyle(
            fontSize: 16.sp,
            fontWeight: FontWeight.w700,
          ),
        ),
        const Spacer(),
        Text(
          'See all reviews',
          style: TextStyle(
            color: kHintColor,
            fontSize: 14.sp,
            fontWeight: FontWeight.w400,
          ),
        ),
      ],
    ),
    const Divider(),
    Row(
      children: [
        CircleAvatar(
          radius: ScreenUtil().radius(20),
        ),
        Gap(ScreenUtil().setWidth(10)),
        Column(
          mainAxisAlignment: MainAxisAlignment.start,
          children: [
            Text(
              'Alexa',
              style: TextStyle(
                fontSize: 20.sp,
                fontWeight: FontWeight.w400,
              ),
            ),
            Gap(ScreenUtil().setHeight(3)),
            Text(
              '01-06-2021',
              style: TextStyle(
                color: const Color(0xFFABABB7),
                fontSize: 12.sp,
              ),
            ),
          ],
        ),
      ],
    ),
  ],
);

```

```

        fontWeight: FontWeight.w400,
    ),
),
],
),
const Spacer(),
const FiveStars(),
],
),
Row(
mainAxisAlignment: MainAxisAlignment.center,
children: [
Gap(ScreenUtil().setWidth(30)),
SizedBox(
width: 255.w,
height: 100.h,
child: Text(
'I have bought over 10 suits from this seller ...wow ...what a
service ...they have size and customize my suits to a perfect and',
style: TextStyle(
color: kHintColor,
fontSize: 15.sp,
fontWeight: FontWeight.w400,
),
),
),
],
),
],
),
],
);
}
}

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';
import
'package:on_budget/src/modules/customer_modules/home/presentation/widgets/product
_details/sliders/size_sliders.dart';

class SizePredictSection extends StatefulWidget {
const SizePredictSection({super.key});

@Override
_SizePredictSectionState createState() => _SizePredictSectionState();

```

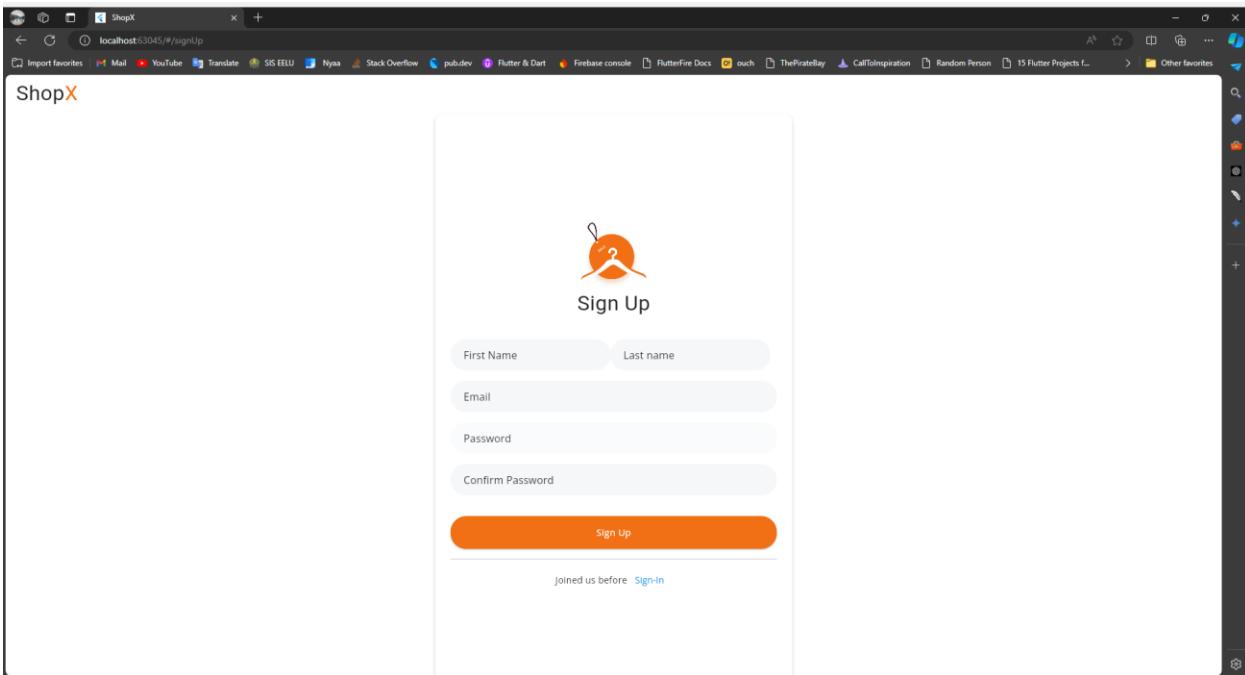
```
}

class _SizePredictSectionState extends State<SizePredictSection> {
    double height = 150;
    double weight = 55;
    double age = 10;

    @override
    Widget build(BuildContext context) {
        return Column(
            children: [
                SizeSliders(
                    valueHeight: height,
                    valueWeight: weight,
                    valueAge: age,
                    onChanged: (newHeight, newWeight, newAge) {
                        setState(() {
                            height = newHeight;
                            weight = newWeight;
                            age = newAge;
                        });
                    },
                ),
                Gap(ScreenUtil().setHeight(20)),
                const Text(
                    'Recommended Size: ',
                    style: TextStyle(fontSize: 24),
                ),
            ],
        );
    }
}
```

## 5.2 Web Application

### 5.2.1 Register



### Code Implementation:

```
import 'package:ecommerce_responsive/screens/screen_home.dart';
import 'package:ecommerce_responsive/screens/screen_sign_in.dart';
import 'package:ecommerce_responsive/utils/widgets/app_widget.dart';
import 'package:ecommerce_responsive/utils/widgets/appbar.dart';
import 'package:ecommerce_responsive/utils/widgets/material_button.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart' hide ContextExtensionss;
import 'package:nb_utils/nb_utils.dart';
import 'package:ecommerce_responsive/utils/colors_constant.dart';
import 'package:ecommerce_responsive/utils/size_constant.dart';
import 'package:ecommerce_responsive/utils/images_constant.dart';
import 'package:ecommerce_responsive/utils/strings_constant.dart';

class ScreenSignUp extends StatefulWidget {
    static const String tag = '/signUp';

    const ScreenSignUp({super.key});
```

```

@Override
ScreenState createState() => ScreenSignUpState();
}

class ScreenSignUpState extends State<ScreenSignUp> {
var emailCont = TextEditingController();
var passwordCont = TextEditingController();
var confirmPasswordCont = TextEditingController();
var firstNameCont = TextEditingController();
var lastNameCont = TextEditingController();

@Override
Widget build(BuildContext context) {
    var width = MediaQuery.of(context).size.width;
    var height = MediaQuery.of(context).size.height;

    return Scaffold(
        appBar: const DefaultAppBar(title: "Sign Up", showActionIcon: false),
        body: SingleChildScrollView(
            child: SizedBox(
                height: context.isPhone()? null: height,
                child: Center(
                    child: Card(
                        elevation: 4,
                        child: ConstrainedBox(
                            constraints: const BoxConstraints(maxWidth: 550),
                            child: Padding(
                                padding: const EdgeInsets.all(24.0),
                                child: Column(
                                    mainAxisAlignment: MainAxisAlignment.center,
                                    children: <Widget>[
                                        ConstrainedBox(
                                            constraints: const BoxConstraints(maxWidth: 100),
                                            child: Image.asset(ic_app_icon, width: width * 0.22)),
                                        Row(
                                            mainAxisSize: MainAxisSize.max,
                                            mainAxisAlignment: MainAxisAlignment.center,
                                            children: <Widget>[
                                                Text("Sign Up", textColor: sh_textColorPrimary,
                                                    fontSize: spacing_xlarge, fontFamily: fontBold),
                                                ],
                                        ),
                                        32.height,
                                        SizedBox(
                                            width: double.infinity,

```

```
        child: LayoutBuilder(
            builder: (BuildContext context, BoxConstraints
constraints) {
                return Wrap(
                    crossAxisAlignment: WrapCrossAlignment.center,
                    children: [
                        Container(
                            width: context.isPhone()? constraints maxWidth:
constraints.maxWidth * 0.49,
                            padding: const EdgeInsets.only(bottom: 16),
                            child: TextFormField(
                                keyboardType: TextInputType.emailAddress,
                                autofocus: false,
                               textInputAction: TextInputAction.next,
                                controller: firstNameCont,
                                textCapitalization: TextCapitalization.words,
                                style: primaryTextStyle(),
                                decoration: InputDecoration(
                                    filled: true,
                                    fillColor: context.cardColor,
                                    focusColor:
sh_editText_background_active,
                                    hintText: sh_hint_first_name,
                                    hintStyle: primaryTextStyle(),
                                    contentPadding: const
EdgeInsets.fromLTRB(20.0, 15.0, 20.0, 15.0),
                                    focusedBorder:
OutlineInputBorder(borderRadius: BorderRadius.circular(32.0), borderSide: const
BorderSide(color: sh_colorPrimary, width: 0.5)),
                                    enabledBorder:
OutlineInputBorder(borderRadius: BorderRadius.circular(32.0), borderSide: const
BorderSide(color: Colors.transparent, style: BorderStyle.none, width: 0))),
                                ),
                                ),
                            ),
                            Container(
                                width: context.isPhone()? constraints maxWidth:
constraints.maxWidth * 0.49,
                                padding: const EdgeInsets.only(bottom: 16),
                                child: TextFormField(
                                    keyboardType: TextInputType.emailAddress,
                                    autofocus: false,
                                    controller: lastNameCont,
                                    textInputAction: TextInputAction.next,
                                    textCapitalization: TextCapitalization.words,
```

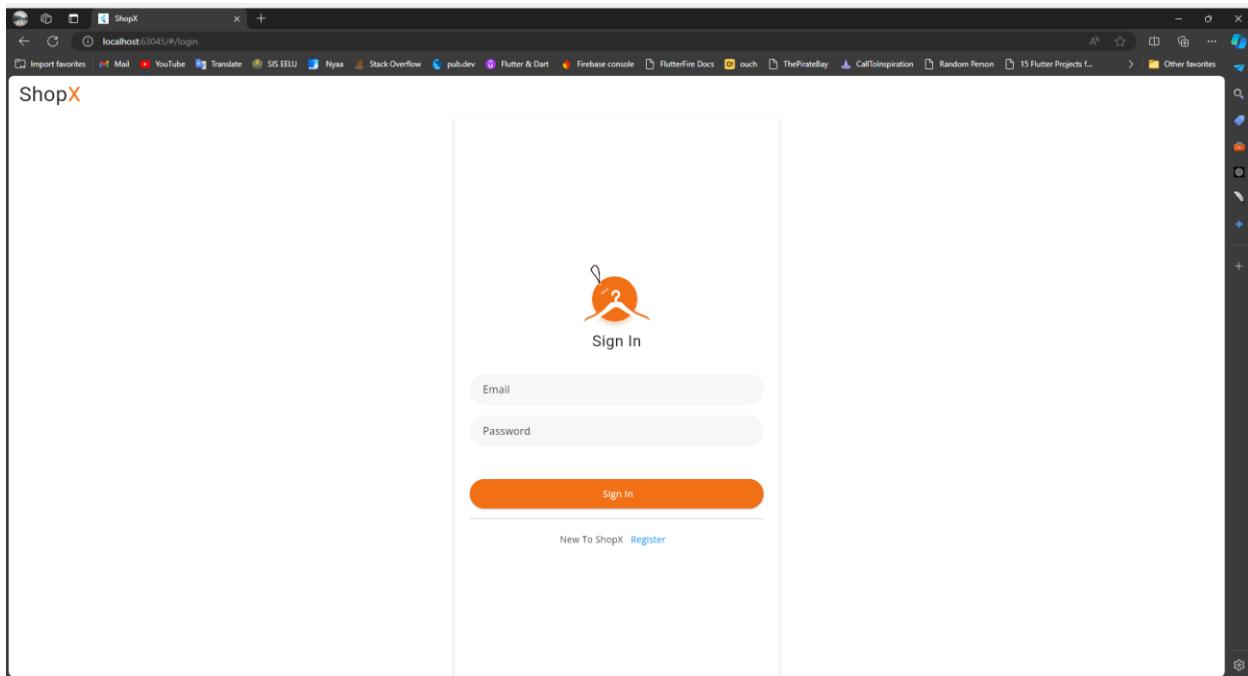
```
        style: primaryTextStyle(),
        decoration: InputDecoration(
            filled: true,
            fillColor: context.cardColor,
            focusColor:
sh_editText_background_active,
            hintText: sh_hint_last_name,
            hintStyle: primaryTextStyle(),
            contentPadding: const
EdgeInsets.fromLTRB(20.0, 15.0, 20.0, 15.0),
            focusedBorder:
OutlineInputBorder(borderRadius: BorderRadius.circular(32.0), borderSide: const
BorderSide(color: sh_colorPrimary, width: 0.5)),
            enabledBorder:
OutlineInputBorder(borderRadius: BorderRadius.circular(32.0), borderSide: const
BorderSide(color: Colors.transparent, style: BorderStyle.none, width: 0))),
        ),
        ],
    );
}
),
),
),
//16.height,
 TextFormField(
    keyboardType: TextInputType.emailAddress,
    autofocus: false,
    controller: emailCont,
   textInputAction: TextInputAction.next,
    textCapitalization: TextCapitalization.words,
    style: primaryTextStyle(),
    decoration: InputDecoration(
        filled: true,
        fillColor: context.cardColor,
        focusColor: sh_editText_background_active,
        hintText: sh_hint_Email,
        hintStyle: primaryTextStyle(),
        contentPadding: const EdgeInsets.fromLTRB(20.0, 15.0,
20.0, 15.0),
        focusedBorder: OutlineInputBorder(borderRadius:
BorderRadius.circular(32.0), borderSide: const BorderSide(color: sh_colorPrimary,
width: 0.5)),
        enabledBorder: OutlineInputBorder(borderRadius:
BorderRadius.circular(32.0), borderSide: const BorderSide(color:
Colors.transparent, style: BorderStyle.none, width: 0))),
```

```
        ),
        16.height,
        TextFormField(
            keyboardType: TextInputType.text,
            autofocus: false,
            obscureText: true,
            textInputAction: TextInputAction.next,
            style: primaryTextStyle(),
            controller: passwordCont,
            textCapitalization: TextCapitalization.words,
            decoration: InputDecoration(
                filled: true,
                fillColor: context.cardColor,
                focusColor: sh_editText_background_active,
                hintStyle: primaryTextStyle(),
                hintText: sh_hint_password,
                contentPadding: const EdgeInsets.fromLTRB(20.0, 15.0,
20.0, 15.0),
                focusedBorder: OutlineInputBorder(borderRadius:
BorderRadius.circular(32.0), borderSide: const BorderSide(color: sh_colorPrimary,
width: 0.5)),
                enabledBorder: OutlineInputBorder(borderRadius:
BorderRadius.circular(32.0), borderSide: const BorderSide(color:
Colors.transparent, style: BorderStyle.none, width: 0))),
            ),
        16.height,
        TextFormField(
            keyboardType: TextInputType.text,
            autofocus: false,
            obscureText: true,
            style: primaryTextStyle(),
            controller: confirmPasswordCont,
            textCapitalization: TextCapitalization.words,
            decoration: InputDecoration(
                filled: true,
                fillColor: context.cardColor,
                focusColor: sh_editText_background_active,
                hintStyle: primaryTextStyle(),
                hintText: sh_hint_confirm_password,
                contentPadding: const EdgeInsets.fromLTRB(20.0, 15.0,
20.0, 15.0),
                focusedBorder: OutlineInputBorder(borderRadius:
BorderRadius.circular(32.0), borderSide: const BorderSide(color: sh_colorPrimary,
width: 0.5)),
```



```
        ),
    );
}
}
```

## 5.2.2 Login



## Code Implementation:

```
import 'package:ecommerce_responsive/screens/screen_home.dart';
import 'package:ecommerce_responsive/utils/widgets/app_widget.dart';
import 'package:ecommerce_responsive/utils/widgets/appbar.dart';
import 'package:ecommerce_responsive/utils/widgets/material_button.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart' hide ContextExtensionss;
import 'package:nb_utils/nb_utils.dart';
import 'package:ecommerce_responsive/screens/screen_sign_up.dart';
import 'package:ecommerce_responsive/utils/colors_constant.dart';
import 'package:ecommerce_responsive/utils/size_constant.dart';
import 'package:ecommerce_responsive/utils/images_constant.dart';
import 'package:ecommerce_responsive/utils/strings_constant.dart';

class ScreenSignIn extends StatefulWidget {
```

```

static const String tag = '/login';

const ScreenSignIn({super.key});

@Override
ScreenState createState() => ScreenSignInState();
}

class ScreenSignInState extends State<ScreenSignIn> {
var emailCont = TextEditingController();
var passwordCont = TextEditingController();
final formKey = GlobalKey<FormState>();

@Override
Widget build(BuildContext context) {
    var width = MediaQuery.of(context).size.width;
    var height = MediaQuery.of(context).size.height;
    return Scaffold(
        appBar: const DefaultAppBar(title: "Sign-In", showActionIcon: false, ),
        body: SizedBox(
            height: height,
            child: SingleChildScrollView(
                child: SizedBox(
                    height: context.isPhone()? null: height,
                    child: Center(
                        child: Card(
                            elevation: 4,
                            child: ConstrainedBox(
                                constraints: const BoxConstraints(maxWidth: 500),
                                child: Padding(
                                    padding: const EdgeInsets.all(24.0),
                                    child: Column(
                                        mainAxisAlignment: MainAxisAlignment.center,
                                        children: <Widget>[
                                            ConstrainedBox(
                                                constraints: const BoxConstraints(maxWidth: 100),
                                                child: Image.asset(ic_app_icon, width: width *
0.22)),
                                            Row(
                                                mainAxisAlignment: MainAxisAlignment.center,
                                                children: <Widget>[
                                                    text("Sign In", textColor: sh_textColorPrimary,
fontSize: spacing_large, fontFamily: fontBold),
                                                ],
                                            ),
                                        ],
                                    ),
                                ),
                            ),
                        ),
                    ),
                ),
            ),
        ),
    );
}
}

```

```
        const SizedBox(height: spacing_xlarge,),  
  
        TextFormField(  
            keyboardType: TextInputType.emailAddress,  
            autofocus: false,  
            controller: emailCont,  
            textInputAction: TextInputAction.next,  
            textCapitalization: TextCapitalization.words,  
            style: primaryTextStyle(),  
            decoration: InputDecoration(  
                filled: true,  
                fillColor: context.cardColor,  
                focusColor: sh_editText_background_active,  
                hintText: sh_hint_Email,  
                hintStyle: primaryTextStyle(),  
                contentPadding: const EdgeInsets.fromLTRB(20.0,  
15.0, 20.0, 15.0),  
                focusedBorder: OutlineInputBorder(borderRadius:  
BorderRadius.circular(32.0), borderSide: const BorderSide(color: sh_colorPrimary,  
width: 0.5)),  
                enabledBorder: OutlineInputBorder(borderRadius:  
BorderRadius.circular(32.0), borderSide: const BorderSide(color:  
Colors.transparent, style: BorderStyle.none, width: 0))),  
            ),  
            16.height,  
            TextFormField(  
                keyboardType: TextInputType.text,  
                autofocus: false,  
                obscureText: true,  
                style: primaryTextStyle(),  
                controller: passwordCont,  
                textCapitalization: TextCapitalization.words,  
                decoration: InputDecoration(  
                    filled: true,  
                    fillColor: context.cardColor,  
                    focusColor: sh_editText_background_active,  
                    hintStyle: primaryTextStyle(),  
                    hintText: sh_hint_password,  
                    contentPadding: const EdgeInsets.fromLTRB(20.0,  
15.0, 20.0, 15.0),  
                    focusedBorder: OutlineInputBorder(borderRadius:  
BorderRadius.circular(32.0), borderSide: const BorderSide(color: sh_colorPrimary,  
width: 0.5)),
```

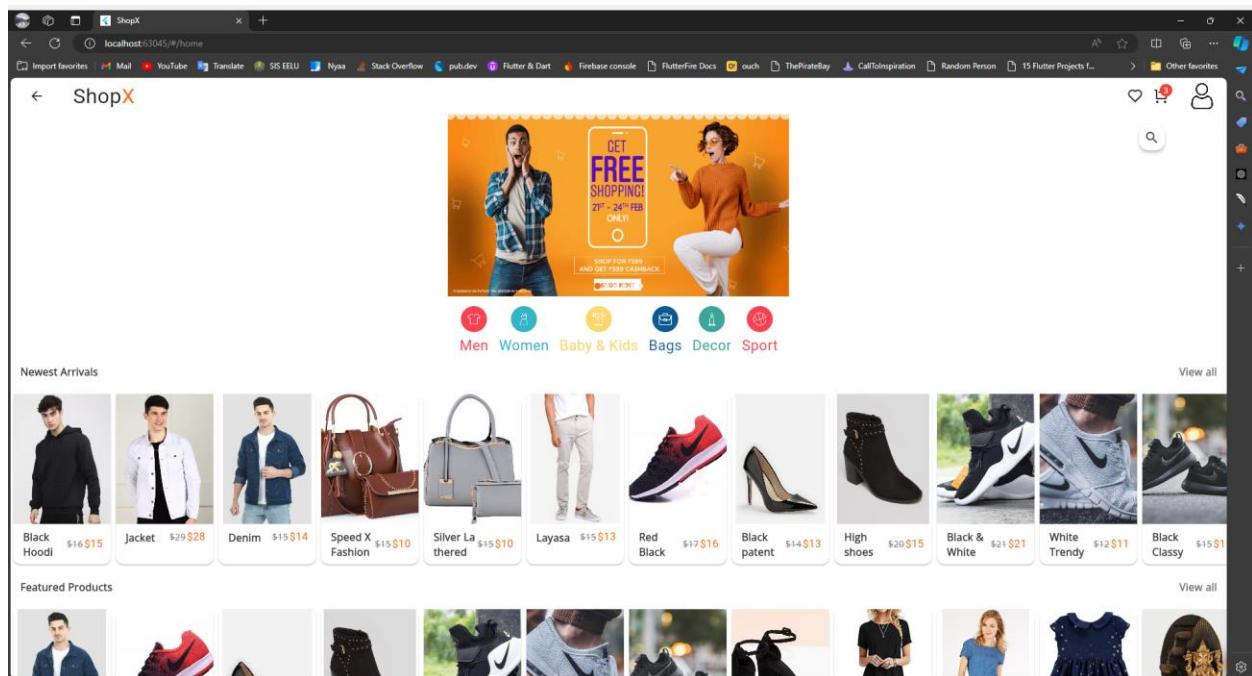
```
enabledBorder: OutlineInputBorder(borderRadius:  
BorderRadius.circular(32.0), borderSide: const BorderSide(color:  
Colors.transparent, style: BorderStyle.none, width: 0))),  
,  
  
50.height,  
  
SizedBox(  
    width: double.infinity,  
    height: 45,  
    // height: double.infinity,  
    child: RoundedButton(  
        title: sh_lbl_sign_in,  
        color: sh_colorPrimary,titleColor: sh_white,  
        onPressed: (){  
            if(kIsWeb) {  
                Get.until((route) => route.settings.name ==  
"/");  
                Get.toNamed(ScreenHome.tag);  
            }  
            else{  
                Get.until((route) => route.settings.name ==  
ScreenHome.tag);  
            }  
        }  
    },)  
,  
8.height,  
const Divider(),  
8.height,  
  
Row(  
    mainAxisAlignment: MainAxisAlignment.center,  
    children: [  
        const Text("New To ShopX"),  
        TextButton(  
            onPressed: ()=> Get.toNamed(ScreenSignUp.tag),  
            child: const Text("Register", style:  
TextStyle(color: Colors.blue),)),  
        ],  
    ),  
    16.height,  
    [,  
    ],  
),  
,
```

```

        ),
        ),
        ),
        ),
        );
    }
}

```

## 5.2.3 Home



## Code Implementation:

```

import 'package:ecommerce_responsive/screens/screen_search_product.dart';
import 'package:ecommerce_responsive/screens/side_drawer.dart';
import
'package:ecommerce_responsive/utils/widgets/dots_indicator/src/dots_decorator.dart';
import
'package:ecommerce_responsive/utils/widgets/dots_indicator/src/dots_indicator.dart';
import 'package:ecommerce_responsive/utils/widgets/footer.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart' hide ContextExtensionss;

```

```

import 'package:nb_utils/nb_utils.dart';
import 'package:ecommerce_responsive/models/model_category.dart';
import 'package:ecommerce_responsive/models/model_product.dart';
import 'package:ecommerce_responsive/screens/screen_category.dart';
import 'package:ecommerce_responsive/screens/screen_view_all_products.dart';
import 'package:ecommerce_responsive/utils/colors_constant.dart';
import 'package:ecommerce_responsive/utils/size_constant.dart';
import 'package:ecommerce_responsive/utils/root_bundle.dart';
import 'package:ecommerce_responsive/utils/strings_constant.dart';
import 'package:ecommerce_responsive/utils/common_widget.dart';
import 'package:ecommerce_responsive/utils/widgets/app_widget.dart';

import '../utils/widgets/appbar.dart';

class ScreenHome extends StatefulWidget {
    static const String tag = '/home';

    const ScreenHome({super.key});

    @override
    ScreenHomeState createState() => ScreenHomeState();
}

class ScreenHomeState extends State<ScreenHome> {
    List<ModelCategory> categoryList = [];
    List<String> banners = [];
    List<ModelProduct> newestProducts = [];
    List<ModelProduct> featuredProducts = [];
    var position = 0;
    var colors = [sh_cat_1, sh_cat_2, sh_cat_3, sh_cat_4, sh_cat_5];

    @override
    void initState() {
        super.initState();
        fetchData();
    }

    fetchData() async {
        loadCategory().then((categories) {
            setState(() {
                categoryList.clear();
                categoryList.addAll(categories);
            });
        }).catchError((error) {
            Toasty(context, error);
        });
    }
}

```

```

});  

List<ModelProduct> products = await loadProducts();  

List<ModelProduct> featured = [];  

products.forEach((product) {  

    if (product.featured!) {  

        featured.add(product);  

    }  

});  

List<String> banner = [];  

for (var i = 1; i < 7; i++) {  

    banner.add("assets/images/shopx/img/products/banners/b$i.jpg");  

}  

setState(() {  

    newestProducts.clear();  

    featuredProducts.clear();  

    banners.clear();  

    banners.addAll(banner);  

    newestProducts.addAll(products);  

    featuredProducts.addAll(featured);  

});  

}  

@Override  

Widget build(BuildContext context) {  

    return Scaffold(  

        appBar: const DefaultAppBar(title: "ShopX", showLeading: true,),  

        drawer: context.isPhone()? const SideDrawer():null,  

        body: newestProducts.isNotEmpty  

            ? SafeArea(  

                child: SingleChildScrollView(  

                    child: Column(  

                        children: <Widget>[  

                            Stack(  

                                alignment: const Alignment(0.9,-0.9),  

                                children: [  

                                    SizedBox(  

                                        child: ConstrainedBox(  

                                            constraints: const BoxConstraints(maxHeight: 280,  

minWidth: 500),  

                                        child: StatefulBuilder(  

                                            builder: (context, setState) {  

                                                return Stack(  

                                                    alignment: Alignment.bottomCenter,  

                                                    children: <Widget>[

```



```

///Category list
if(!context.isPhone())
Center(
    child: ListView.builder(
        itemCount: categoryList.length,
        shrinkWrap: true,
        scrollDirection: Axis.horizontal,
        padding: const EdgeInsets.only(left: spacing_standard,
right: spacing_standard,top: spacing_standard_new),
        itemBuilder: (BuildContext context, int index) {
            return Container(
                margin: const EdgeInsets.only(left: spacing_standard,
right: spacing_standard),
                child: Column(
                    children: <Widget>[
                        Container(
                            padding: const EdgeInsets.all(spacing_middle),
                            decoration: BoxDecoration(shape:
BoxShape.circle, color: colors[index % colors.length]),
                            child: Image.asset(categoryList[index].image!,
width: 20, color: sh_white),
                        ),
                        const SizedBox(height: spacing_control),
                        text(categoryList[index].name, textColor:
colors[index % colors.length], fontFamily: fontMedium,fontSize: 22)
                    ],
                ),
            ).onTap(() {
                ScreenCategory(category:
categoryList[index]).launch(context);
            });
        },
    ),
).withHeight(100),

///Newest Product
horizontalHeading(sh_lbl_newest_product, callback: () {
    ScreenViewAllProduct(products: newestProducts, title:
sh_lbl_newest_product).launch(context);
}),
ProductHorizontalList(newestProducts),

const SizedBox(height: spacing_standard_new),
horizontalHeading(sh_lbl_Featured, callback: () {

```

```

        ScreenViewAllProduct(products: featuredProducts, title:
sh_lbl_Featured).launch(context);
    },
    ProductHorizontalList(featuredProducts),
    const SizedBox(height: 60),

        const Footer()
    ],
),
),
),
)
: Container(),
);
}
}

import 'package:ecommerce_responsive/utils/url_extension.dart';
import 'package:ecommerce_responsive/utils/images_constant.dart';
import 'package:ecommerce_responsive/utils/extension/currency_extension.dart';
import 'package:ecommerce_responsive/utils/widgets/app_widget.dart';
import 'package:ecommerce_responsive/utils/widgets/appbar.dart';
import 'package:ecommerce_responsive/utils/widgets/image_holder.dart';
import 'package:flutter/material.dart';
import 'package:flutter_layout_grid/flutter_layout_grid.dart';
import 'package:get/get.dart' hide ContextExtensionss;
import 'package:nb_utils/nb_utils.dart';
import 'package:ecommerce_responsive/models/model_product.dart';
import 'package:ecommerce_responsive/utils/colors_constant.dart';
import 'package:ecommerce_responsive/utils/size_constant.dart';
import 'package:ecommerce_responsive/utils/root_bundle.dart';
import 'package:ecommerce_responsive/utils/common_widget.dart';
import 'package:ecommerce_responsive/main.dart';
import 'package:ecommerce_responsive/utils/widgets/flutter_rating_bar.dart';

import 'screen_product_detail.dart';

class ScreenSearchProduct extends StatefulWidget {
    static const String tag = '/search';

    const ScreenSearchProduct({super.key});

    @override
    ScreenSearchProductState createState() => ScreenSearchProductState();
}

```

```

class ScreenSearchProductState extends State<ScreenSearchProduct> {
    TextEditingController searchController = TextEditingController();
    List<ModelProduct> resultList = [];
    bool isLoadingMoreData = false;
    bool isEmpty = false;
    var searchText = "";
    List<ModelProduct> products = [];
    List<TrackSize> rowSizes = [];

    @override
    void initState() {
        super.initState();
        fetchData();
    }

    fetchData() async {
        products = await loadProducts();
    }

    searchData() async {
        List<ModelProduct> filteredList = [];
        for (var product in products) {
            if (product.name!.toLowerCase().contains(searchText)) {
                filteredList.add(product);
            }
        }
        setState(() {
            resultList.clear();
            resultList.addAll(filteredList);
            isEmpty = resultList.isEmpty;
            rowSizes = List.filled(resultList.length, auto);
        });
    }

    @override
    Widget build(BuildContext context) {

        Widget searchList = resultList.isNotEmpty?
        LayoutGrid(
            columnSizes: context.isPhone() ? [1.fr] : [1.fr, 1.fr],
            rowSizes: rowSizes,
            rowGap: 15,
            columnGap: 15,
            children: [
                for (var index = 0; index < resultList.length; index++)

```

```

InkWell(
    onTap: () {
        Get.toNamed(ScreenProductDetail.tag, parameters:
resultList[index]
            .toJson()
            .encode);
    },
    child: Container(
        padding: const EdgeInsets.all(10.0),
        constraints: const BoxConstraints(maxHeight: 200),
        child: Row(
            children: <Widget>[
                ImageHolder(
                    imagePath:
"${base}img/products${resultList[index].images![0]
            .src!}"),
                const SizedBox(width: 10),
                Expanded(
                    child: Column(
                        mainAxisAlignment: MainAxisAlignment.start,
                        children: <Widget>[
                            text(resultList[index].name, textColor:
sh_textColorPrimary),
                            const SizedBox(height: 4),
                            Row(
                                children: <Widget>[
                                    text(resultList[index].on_sale! ? resultList[index]
                                        .sale_price.toString()
                                        .toCurrencyFormat() : resultList[index].price
                                        .toString().toCurrencyFormat(),
                                    textColor: sh_colorPrimary,
                                    fontFamily: fontMedium,
                                    fontSize: textSizeNormal),
                                    const SizedBox(width: spacing_control,),

                                    Text(
                                        resultList[index].regular_price.toString()
                                            .toCurrencyFormat()!,
                                        style: const TextStyle(color:
sh_textColorSecondary,
                                            fontFamily: fontRegular,
                                            fontSize: textSizeSmall,
                                            decoration: TextDecoration.lineThrough),

```

```
        ),
    ],
),

const SizedBox(height: spacing_standard,),

Row(children:
colorWidget(resultList[index].attributes!)),


const SizedBox(height: 4),


Flexible(child: Text(
resultList[index].description!, maxLines: 3,
overflow: TextOverflow.ellipsis,)),


Row(
mainAxisAlignment: MainAxisAlignment.spaceBetween,
children: <Widget>[
RatingBar(
initialRating: double.parse(
resultList[index].average_rating!), direction: Axis.horizontal,
allowHalfRating: true,
tapOnlyMode: true,
itemCount: 5,
itemSize: 16,
itemBuilder: (context, _) =>
const Icon(
Icons.star,
color: Colors.amber,
),
onRatingUpdate: (rating) {},),
Container(
padding: const EdgeInsets.all(spacing_control),
margin: const EdgeInsets.only(right:
spacing_standard),
decoration: const BoxDecoration(
shape: BoxShape.circle, color: sh_white),
child: const Icon(
Icons.favorite_border,
color: sh_textColorPrimary,
size: 16,
),
)
)
```





```

        }

    }

import 'package:flutter/material.dart';
import 'package:get/get_utils/get_utils.dart';

class ImageHolder extends StatelessWidget {
    const ImageHolder({Key? key, required this.imagePath}) : super(key: key);
    final String imagePath;
    @override
    Widget build(BuildContext context) {
        return ConstrainedBox(
            constraints: const BoxConstraints(maxWidth: 180, minWidth: 90),
            child: Image.asset(
                imagePath,
                width: context.width * 0.25,
                height: context.width * 0.3,
                fit: BoxFit.contain,
            ),
        );
    }
}

import 'model_category.dart';

class ModelProduct {
    int? id;
    String? name;

    // ignore: non_constant_identifier_names
    String? date_created;

    // ignore: non_constant_identifier_names
    String? date_created_gmt;

    // ignore: non_constant_identifier_names
    String? date_modified;

    // ignore: non_constant_identifier_names
    String? date_modified_gmt;
    String? type;
    String? status;
    bool? featured;
}

```

```
// ignore: non_constant_identifier_names
String? catalog_visibility;
String? description;

// ignore: non_constant_identifier_names
String? short_description;
String? sku;
String? price;

// ignore: non_constant_identifier_names
String? regular_price;

// ignore: non_constant_identifier_names
String? sale_price;

// ignore: non_constant_identifier_names
String? price_html;

// ignore: non_constant_identifier_names
bool? on_sale;
bool? purchasable;

// ignore: non_constant_identifier_names
int? total_sales;
bool? virtual;
bool? downloadable;

// ignore: non_constant_identifier_names
String? external_url;

// ignore: non_constant_identifier_names
String? button_text;

// ignore: non_constant_identifier_names
String? tax_status;

// ignore: non_constant_identifier_names
String? tax_class;

// ignore: non_constant_identifier_names
bool? manage_stock;

// ignore: non_constant_identifier_names
int? stock_quantity;
```

```
// ignore: non_constant_identifier_names
String? stock_status;
String? backorders;

// ignore: non_constant_identifier_names
bool? backorders_allowed;
bool? backordered;

// ignore: non_constant_identifier_names
bool? sold_individually;
String? weight;
Dimensions? dimensions;

// ignore: non_constant_identifier_names
bool? shipping_required;

// ignore: non_constant_identifier_names
bool? shipping_taxable;

// ignore: non_constant_identifier_names
String? shipping_class;

// ignore: non_constant_identifier_names
int? shipping_class_id;

// ignore: non_constant_identifier_names
bool? reviews_allowed;

// ignore: non_constant_identifier_names
String? average_rating;

// ignore: non_constant_identifier_names
int? rating_count;

// ignore: non_constant_identifier_names
List<dynamic>? related_ids;

// ignore: non_constant_identifier_names
List<dynamic>? upsell_ids;

// ignore: non_constant_identifier_names
List<dynamic>? cross_sell_ids;

// ignore: non_constant_identifier_names
```

```
int? parent_id;

// ignore: non_constant_identifier_names
String? purchase_note;
List<ModelCategory>? categories;
List<dynamic>? tags;
List<ShImage>? images;
List<Attribute>? attributes;

// ignore: non_constant_identifier_names
List<dynamic>? default_attributes;

ModelProduct(
    {this.id,
    this.name,
    // ignore: non_constant_identifier_names
    this.date_created,
    // ignore: non_constant_identifier_names
    this.date_created_gmt,
    // ignore: non_constant_identifier_names
    this.date_modified,
    // ignore: non_constant_identifier_names
    this.date_modified_gmt,
    this.type,
    this.status,
    this.featured,
    // ignore: non_constant_identifier_names
    this.catalog_visibility,
    this.description,
    // ignore: non_constant_identifier_names
    this.short_description,
    this.sku,
    this.price,
    // ignore: non_constant_identifier_names
    this.regular_price,
    // ignore: non_constant_identifier_names
    this.sale_price,
    // ignore: non_constant_identifier_names
    this.price_html,
    // ignore: non_constant_identifier_names
    this.on_sale,
    this.purchasable,
    // ignore: non_constant_identifier_names
    this.total_sales,
    this.virtual,
```

```
this.downloadable,  
  
    // ignore: non_constant_identifier_names  
this.external_url,  
    // ignore: non_constant_identifier_names  
this.button_text,  
    // ignore: non_constant_identifier_names  
this.tax_status,  
    // ignore: non_constant_identifier_names  
this.tax_class,  
    // ignore: non_constant_identifier_names  
this.manage_stock,  
    // ignore: non_constant_identifier_names  
this.stock_quantity,  
    // ignore: non_constant_identifier_names  
this.stock_status,  
this.backorders,  
    // ignore: non_constant_identifier_names  
this.backorders_allowed,  
this.backordered,  
    // ignore: non_constant_identifier_names  
this.sold_individually,  
this.weight,  
this.dimensions,  
    // ignore: non_constant_identifier_names  
this.shipping_required,  
    // ignore: non_constant_identifier_names  
this.shipping_taxable,  
    // ignore: non_constant_identifier_names  
this.shipping_class,  
    // ignore: non_constant_identifier_names  
this.shipping_class_id,  
    // ignore: non_constant_identifier_names  
this.reviews_allowed,  
    // ignore: non_constant_identifier_names  
this.average_rating,  
    // ignore: non_constant_identifier_names  
this.rating_count,  
    // ignore: non_constant_identifier_names  
this.related_ids,  
    // ignore: non_constant_identifier_names  
this.upsell_ids,  
    // ignore: non_constant_identifier_names  
this.cross_sell_ids,  
    // ignore: non_constant_identifier_names
```

```

    this.parent_id,
    // ignore: non_constant_identifier_names
    this.purchase_note,
    this.categories,
    this.tags,
    this.images,
    this.attributes,
    // ignore: non_constant_identifier_names
    this.default_attributes});

factory ModelProduct.fromJson(Map<String, dynamic> json) {
  return ModelProduct(
    id: json['id'],
    name: json['name'],
    date_created: json['date_created'],
    date_created_gmt: json['date_created_gmt'],
    date_modified: json['date_modified'],
    date_modified_gmt: json['date_modified_gmt'],
    type: json['type'],
    status: json['status'],
    featured: json['featured'],
    catalog_visibility: json['catalog_visibility'],
    description: json['description'],
    short_description: json['short_description'],
    sku: json['sku'],
    price: json['price'],
    regular_price: json['regular_price'],
    sale_price: json['sale_price'],
    price_html: json['price_html'],
    on_sale: json['on_sale'],
    purchasable: json['purchasable'],
    total_sales: json['total_sales'],
    virtual: json['virtual'],
    downloadable: json['downloadable'],
    external_url: json['external_url'],
    button_text: json['button_text'],
    tax_status: json['tax_status'],
    tax_class: json['tax_class'],
    manage_stock: json['manage_stock'],
    stock_quantity: json['stock_quantity'],
    stock_status: json['stock_status'],
    backorders: json['backorders'],
    backorders_allowed: json['backorders_allowed'],
    backordered: json['backordered'],
    sold_individually: json['sold_individually'],
  );
}

```

```

        weight: json['weight'],
        dimensions: json['dimensions'] != null ?
Dimensions.fromJson(json['dimensions']) : null,
        shipping_required: json['shipping_required'],
        shipping_taxable: json['shipping_taxable'],
        shipping_class: json['shipping_class'],
        shipping_class_id: json['shipping_class_id'],
        reviews_allowed: json['reviews_allowed'],
        average_rating: json['average_rating'],
        rating_count: json['rating_count'],
        parent_id: json['parent_id'],
        purchase_note: json['purchase_note'],
        categories: json['categories'] != null ? (json['categories'] as
List).map((i) => ModelCategory.fromJson(i)).toList() : null,
        images: json['images'] != null ? (json['images'] as List).map((i) =>
ShImage.fromJson(i)).toList() : null,
        attributes: json['attributes'] != null ? (json['attributes'] as
List).map((i) => Attribute.fromJson(i)).toList() : null,
    );
}

Map<String, dynamic> toJson() {
    final Map<String, dynamic> data = <String, dynamic>{};
    data['id'] = id;
    data['name'] = name;
    data['date_created'] = date_created;
    data['date_created_gmt'] = date_created_gmt;
    data['date_modified'] = date_modified;
    data['date_modified_gmt'] = date_modified_gmt;
    data['type'] = type;
    data['status'] = status;
    data['featured'] = featured;
    data['catalog_visibility'] = catalog_visibility;
    data['description'] = description;
    data['short_description'] = short_description;
    data['sku'] = sku;
    data['price'] = price;
    data['regular_price'] = regular_price;
    data['sale_price'] = sale_price;
    data['price_html'] = price_html;
    data['on_sale'] = on_sale;
    data['purchasable'] = purchasable;
    data['total_sales'] = total_sales;
    data['virtual'] = virtual;
    data['downloadable'] = downloadable;
}

```

```

data['external_url'] = external_url;
data['button_text'] = button_text;
data['tax_status'] = tax_status;
data['tax_class'] = tax_class;
data['manage_stock'] = manage_stock;
data['stock_quantity'] = stock_quantity;
data['stock_status'] = stock_status;
data['backorders'] = backorders;
data['backorders_allowed'] = backorders_allowed;
data['backordered'] = backordered;
data['sold_individually'] = sold_individually;
data['weight'] = weight;
if (dimensions != null) {
    data['dimensions'] = dimensions!.toJson();
}
data['shipping_required'] = shipping_required;
data['shipping_taxable'] = shipping_taxable;
data['shipping_class'] = shipping_class;
data['shipping_class_id'] = shipping_class_id;
data['reviews_allowed'] = reviews_allowed;
data['average_rating'] = average_rating;
data['rating_count'] = rating_count;

data['parent_id'] = parent_id;
data['purchase_note'] = purchase_note;
if (categories != null) {
    data['categories'] = categories!.map((v) => v.toJson()).toList();
}
if (tags != null) {
    data['tags'] = tags!.map((v) => v.toJson()).toList();
}
if (images != null) {
    data['images'] = images!.map((v) => v.toJson()).toList();
}
if (attributes != null) {
    data['attributes'] = attributes!.map((v) => v.toJson()).toList();
}
if (default_attributes != null) {
    data['default_attributes'] =
        default_attributes!.map((v) => v.toJson()).toList();
}
return data;
}
}

```

```

class ShImage {
    int? id;

    // ignore: non_constant_identifier_names
    String? date_created;

    // ignore: non_constant_identifier_names
    String? date_created_gmt;

    // ignore: non_constant_identifier_names
    String? date_modified;

    // ignore: non_constant_identifier_names
    String? date_modified_gmt;
    String? src;
    String? name;
    String? alt;

    // ignore: non_constant_identifier_names
    ShImage({this.id, this.date_created, this.date_created_gmt, this.date_modified,
    this.date_modified_gmt, this.src, this.name, this.alt});

    factory ShImage.fromJson(Map<String, dynamic> json) {
        return ShImage(
            id: json['id'],
            date_created: json['date_created'],
            date_created_gmt: json['date_created_gmt'],
            date_modified: json['date_modified'],
            date_modified_gmt: json['date_modified_gmt'],
            src: json['src'],
            name: json['name'],
            alt: json['alt'],
        );
    }

    Map<String, dynamic> toJson() {
        final Map<String, dynamic> data = <String, dynamic>{};
        data['id'] = id;
        data['date_created'] = date_created;
        data['date_created_gmt'] = date_created_gmt;
        data['date_modified'] = date_modified;
        data['date_modified_gmt'] = date_modified_gmt;
        data['src'] = src;
        data['name'] = name;
        data['alt'] = alt;
    }
}

```

```

        return data;
    }
}

class Attribute {
    int? id;
    String? name;
    int? position;
    bool? visible;
    bool? variation;
    List<String>? options;

    Attribute({this.id, this.name, this.position, this.visible, this.variation,
    this.options});

    factory Attribute.fromJson(Map<String, dynamic> json) {
        return Attribute(
            id: json['id'],
            name: json['name'],
            position: json['position'],
            visible: json['visible'],
            variation: json['variation'],
            options: json['options'] != null ? List<String>.from(json['options']) :
null,
        );
    }
}

Map<String, dynamic> toJson() {
    final Map<String, dynamic> data = <String, dynamic>{};
    data['id'] = id;
    data['name'] = name;
    data['position'] = position;
    data['visible'] = visible;
    data['variation'] = variation;
    if (options != null) {
        data['options'] = options;
    }
    return data;
}
}

class Dimensions {
    String? length;
    String? width;
    String? height;
}

```

```

Dimensions({this.length, this.width, this.height});

factory Dimensions.fromJson(Map<String, dynamic> json) {
  return Dimensions(
    length: json['length'],
    width: json['width'],
    height: json['height'],
  );
}

Map<String, dynamic> toJson() {
  final Map<String, dynamic> data = <String, dynamic>{};
  data['length'] = length;
  data['width'] = width;
  data['height'] = height;
  return data;
}
}

import 'package:flutter/material.dart';

/// Defines widgets which are to used as rating bar items.
class RatingWidget {
  /// Defines widget to be used as rating bar item when the item is completely
  rated.
  final Widget full;

  /// Defines widget to be used as rating bar item when only the half portion of
  item is rated.
  final Widget half;

  /// Defines widget to be used as rating bar item when the item is unrated.
  final Widget empty;

  RatingWidget({
    required this.full,
    required this.half,
    required this.empty,
  });
}

class _HalfRatingWidget extends StatelessWidget {
  final Widget child;
  final double size;

```

```
final bool enableMask;
final bool rtlMode;
final Color? unratedColor;

const _HalfRatingWidget({
    required this.size,
    required this.child,
    required this.enableMask,
    required this.rtlMode,
    required this.unratedColor,
});

@Override
Widget build(BuildContext context) {
    return SizedBox(
        height: size,
        width: size,
        child: enableMask
            ? Stack(
                fit: StackFit.expand,
                children: [
                    FittedBox(
                        fit: BoxFit.contain,
                        child: _NoRatingWidget(
                            size: size,
                            unratedColor: unratedColor,
                            enableMask: enableMask,
                            child: child,
                        ),
                    ),
                    FittedBox(
                        fit: BoxFit.contain,
                        child: ClipRect(
                            clipper: _HalfClipper(
                                rtlMode: rtlMode,
                            ),
                            child: child,
                        ),
                    ),
                ],
            )
            : FittedBox(
                fit: BoxFit.contain,
                child: child,
            ),
    );
}
```

```

    );
}
}

class _HalfClipper extends CustomClipper<Rect> {
    final bool rtlMode;

    _HalfClipper({
        required this.rtlMode,
    });

    @override
    Rect getClip(Size size) => rtlMode
        ? Rect.fromLTRB(
            size.width / 2,
            0.0,
            size.width,
            size.height,
        )
        : Rect.fromLTRB(
            0.0,
            0.0,
            size.width / 2,
            size.height,
        );
}

@Override
bool shouldReclip(CustomClipper<Rect> oldClipper) => true;
}

class _NoRatingWidget extends StatelessWidget {
    final double size;
    final Widget child;
    final bool enableMask;
    final Color? unratedColor;

    const _NoRatingWidget({
        required this.size,
        required this.child,
        required this.enableMask,
        required this.unratedColor,
    });

    @override
    Widget build(BuildContext context) {

```

```

        return SizedBox(
            height: size,
            width: size,
            child: FittedBox(
                fit: BoxFit.contain,
                child: enableMask
                    ? _ColorFilter(
                        color: unratedColor,
                        child: child,
                    )
                    : child,
            ),
        );
    }
}

class _ColorFilter extends StatelessWidget {
    final Widget child;
    final Color? color;

    const _ColorFilter({
        required this.child,
        required this.color,
    });

    @override
    Widget build(BuildContext context) {
        return ColorFiltered(
            colorFilter: ColorFilter.mode(
                color!,
                BlendMode.srcATop,
            ),
            child: ColorFiltered(
                colorFilter: const ColorFilter.mode(
                    Colors.white,
                    BlendMode.srcATop,
                ),
                child: child,
            ),
        );
    }
}

class _IndicatorClipper extends CustomClipper<Rect> {
    final double? ratingFraction;

```

```

final bool rtlMode;

IndicatorClipper({
  this.ratingFraction,
  this.rtlMode = false,
});

@Override
Rect getClip(Size size) => rtlMode
? Rect.fromLTRB(
  size.width - size.width * ratingFraction!,
  0.0,
  size.width,
  size.height,
)
: Rect.fromLTRB(
  0.0,
  0.0,
  size.width * ratingFraction!,
  size.height,
);

@Override
bool shouldReclip(CustomClipper<Rect> oldClipper) => true;
}

/// A widget to display rating as assigned using [rating] property.
///
/// It's a read only version of [RatingBar].
/// Use [RatingBar], if interative version is required. i.e. if user input is
required.

class RatingBarIndicator extends StatefulWidget {
  /// Defines the rating value for indicator.
  ///
  /// Default = 0.0
  final double rating;

  /// {@macro flutterRatingBar.itemCount}
  final int itemCount;

  /// {@macro flutterRatingBar.itemSize}
  final double itemSize;

  /// {@macro flutterRatingBar.itemPadding}
  final EdgeInsets itemPadding;
}

```

```

    /// Controls the scrolling behaviour of rating bar.
    ///
    /// Default is [NeverScrollableScrollPhysics].
    final ScrollPhysics physics;

    /// {@macro flutterRatingBar.textDirection}
    final TextDirection? textDirection;

    /// {@macro flutterRatingBar.itemBuilder}
    final IndexedWidgetBuilder itemBuilder;

    /// {@macro flutterRatingBar.direction}
    final Axis direction;

    /// {@macro flutterRatingBar.unratedColor}
    final Color? unratedColor;

    const RatingBarIndicator({super.key,
        required this.itemBuilder,
        this.rating = 0.0,
        this.itemCount = 5,
        this.itemSize = 40.0,
        this.itemPadding = const EdgeInsets.all(0.0),
        this.physics = const NeverScrollableScrollPhysics(),
        this.textDirection,
        this.direction = Axis.horizontal,
        this.unratedColor,
    });
}

@Override
RatingBarIndicatorState createState() => RatingBarIndicatorState();
}

class RatingBarIndicatorState extends State<RatingBarIndicator> {
    double _ratingFraction = 0.0;
    int _ratingNumber = 0;
    bool _isRTL = false;

    @override
    void initState() {
        super.initState();
        _ratingNumber = widget.rating.truncate() + 1;
        _ratingFraction = widget.rating - _ratingNumber + 1;
    }
}

```

```

@Override
Widget build(BuildContext context) {
    _isRTL = (widget.textDirection ?? Directionality.of(context)) ==
TextDirection.rtl;
    _ratingNumber = widget.rating.truncate() + 1;
    _ratingFraction = widget.rating - _ratingNumber + 1;
    return SingleChildScrollView(
        scrollDirection: widget.direction,
        physics: widget.physics,
        child: widget.direction == Axis.horizontal
            ? Row(
                mainAxisAlignment: MainAxisAlignment.min,
                textDirection: _isRTL ? TextDirection.rtl : TextDirection.ltr,
                children: _children(),
            )
            : Column(
                mainAxisAlignment: MainAxisAlignment.min,
                textDirection: _isRTL ? TextDirection.rtl : TextDirection.ltr,
                children: _children(),
            ),
    );
}

List<Widget> _children() {
    return List.generate(
        widget.itemCount,
        (index) {
            if (widget.textDirection != null) {
                if (widget.textDirection == TextDirection.rtl &&
Directionality.of(context) != TextDirection.rtl) {
                    return Transform(
                        transform: Matrix4.identity()..scale(-1.0, 1.0, 1.0),
                        alignment: Alignment.center,
                        transformHitTests: false,
                        child: _buildItems(index),
                    );
                }
            }
            return _buildItems(index);
        },
    );
}

Widget _buildItems(int index) => Padding(

```

```

padding: widget.itemPadding,
child: SizedBox(
  width: widget.itemSize,
  height: widget.itemSize,
  child: Stack(
    fit: StackFit.expand,
    children: [
      FittedBox(
        fit: BoxFit.contain,
        child: index + 1 < _ratingNumber
          ? widget.itemBuilder(context, index)
          : _ColorFilter(
              color: widget.unratedColor ?? Colors.grey[200],
              child: widget.itemBuilder(context, index),
            ),
      ),
      if (index + 1 == _ratingNumber)
        _isRTL
          ? FittedBox(
              fit: BoxFit.contain,
              child: ClipRect(
                clipper: _IndicatorClipper(
                  ratingFraction: _ratingFraction,
                  rtlMode: _isRTL,
                ),
                child: widget.itemBuilder(context, index),
              ),
            )
          : FittedBox(
              fit: BoxFit.contain,
              child: ClipRect(
                clipper: _IndicatorClipper(
                  ratingFraction: _ratingFraction,
                ),
                child: widget.itemBuilder(context, index),
              ),
            ),
    ],
  ),
);
}

/// A widget to receive rating input from users.
///

```

```
/// [RatingBar] can also be used to display rating
///
/// Prefer using [RatingBarIndicator] instead, if read only version is required.
/// As RatingBarIndicator supports any fractional rating value.
class RatingBar extends StatefulWidget {
    /// {@template flutterRatingBar.itemCount}
    /// Defines total number of rating bar items.
    ///
    /// Default = 5
    /// {@endtemplate}
    final int itemCount;

    /// Defines the initial rating to be set to the rating bar.
    final double? initialRating;

    /// Return current rating whenever rating is updated.
    final ValueChanged<double> onRatingUpdate;

    /// {@template flutterRatingBar.itemSize}
    /// Defines width and height of each rating item in the bar.
    ///
    /// Default = 40.0
    /// {@endtemplate}
    final double itemSize;

    /// Default [allowHalfRating] = false. Setting true enables half rating
    support.
    final bool allowHalfRating;

    /// {@template flutterRatingBar.itemPadding}
    /// The amount of space by which to inset each rating item.
    /// {@endtemplate}
    final EdgeInsets itemPadding;

    /// if set to true, will disable any gestures over the rating bar.
    ///
    /// Default = false
    final bool ignoreGestures;

    /// if set to true will disable drag to rate feature. Note: Enabling this mode
    will disable half rating capability.
    ///
    /// Default = false
    final bool tapOnlyMode;
```

```
/// {@template flutterRatingBar.textDirection}
/// The text flows from right to left if [textDirection] = TextDirection rtl
/// {@endtemplate}
final TextDirection? textDirection;

/// {@template flutterRatingBar.itemBuilder}
/// Widget for each rating bar item.
/// {@endtemplate}
final IndexedWidgetBuilder? itemBuilder;

/// Customizes the Rating Bar item with [RatingWidget].
final RatingWidget? ratingWidget;

/// if set to true, Rating Bar item will glow when being touched.
///
/// Default = true
final bool glow;

/// Defines the radius of glow.
///
/// Default = 2
final double glowRadius;

/// Defines color for glow.
///
/// Default = theme's accent color
final Color? glowColor;

/// {@template flutterRatingBar.direction}
/// Direction of rating bar.
///
/// Default = Axis.horizontal
/// {@endtemplate}
final Axis direction;

/// {@template flutterRatingBar.unratedColor}
/// Defines color for the unrated portion.
///
/// Default = Colors.grey[200]
/// {@endtemplate}
final Color? unratedColor;

/// Sets minimum rating
///
/// Default = 0
```

```

final double minRating;

/// Sets maximum rating
///
/// Default = [itemCount]
final double? maxRating;

const RatingBar({super.key,
  this.itemCount = 5,
  this.initialRating = 0.0,
  required this.onRatingUpdate,
  this.itemSize = 40.0,
  this.allowHalfRating = false,
  this.itemBuilder,
  this.itemPadding = const EdgeInsets.all(0.0),
  this.ignoreGestures = false,
  this.tapOnlyMode = false,
  this.textDirection,
  this.ratingWidget,
  this.glow = true,
  this.glowRadius = 2,
  this.direction = Axis.horizontal,
  this.glowColor,
  this.unratedColor,
  this.minRating = 0,
  this.maxRating,
}) : assert(
  (itemBuilder == null && ratingWidget != null) || (itemBuilder != null
&& ratingWidget == null),
  'itemBuilder and ratingWidget can\'t be initialized at the same time.'
  'Either remove ratingWidget or itemBuilder.',
);

@Override
RatingBarState createState() => RatingBarState();
}

class RatingBarState extends State<RatingBar> {
double? _rating = 0.0;

//double _ratingHistory = 0.0;
double iconRating = 0.0;
double? _minRating, _maxRating;
bool _isRTL = false;
final ValueNotifier<bool> _glow = ValueNotifier(false);

```

```

@Override
void initState() {
    super.initState();
    _minRating = widget.minRating;
    _maxRating = widget.maxRating ?? widget.itemCount.toDouble();
    _rating = widget.initialRating;
}

@Override
void didUpdateWidget(RatingBar oldWidget) {
    super.didUpdateWidget(oldWidget);
    if (oldWidget.initialRating != widget.initialRating) {
        _rating = widget.initialRating;
    }
    _minRating = widget.minRating;
    _maxRating = widget.maxRating ?? widget.itemCount.toDouble();
}

@Override
void dispose() {
    _glow.dispose();
    super.dispose();
}

@Override
Widget build(BuildContext context) {
    _isRTL = (widget.textDirection ?? Directionality.of(context)) ==
    TextDirection.rtl;
    iconRating = 0.0;
    return Material(
        color: Colors.transparent,
        child: Wrap(
            alignment: WrapAlignment.start,
            textDirection: _isRTL ? TextDirection.rtl : TextDirection.ltr,
            direction: widget.direction,
            children: List.generate(
                widget.itemCount,
                (index) => _buildRating(context, index),
            ),
        ),
    );
}

Widget _buildRating(BuildContext context, int index) {

```

```

Widget ratingWidget;
if (index >= _rating!) {
  ratingWidget = _NoRatingWidget(
    size: widget.itemSize,
    enableMask: widget.ratingWidget == null,
    unratedColor: widget.unratedColor ?? Colors.grey[200],
    child: widget.ratingWidget?.empty ?? widget.itemBuilder!(context, index),
  );
} else if (index >= _rating! - (widget.allowHalfRating ? 0.5 : 1.0) && index < _rating! && widget.allowHalfRating) {
  if (widget.ratingWidget?.half == null) {
    ratingWidget = _HalfRatingWidget(
      size: widget.itemSize,
      enableMask: widget.ratingWidget == null,
      rtlMode: _isRTL,
      unratedColor: widget.unratedColor ?? Colors.grey[200],
      child: widget.itemBuilder!(context, index),
    );
  } else {
    ratingWidget = SizedBox(
      width: widget.itemSize,
      height: widget.itemSize,
      child: FittedBox(
        fit: BoxFit.contain,
        child: _isRTL
          ? Transform(
              transform: Matrix4.identity()..scale(-1.0, 1.0, 1.0),
              alignment: Alignment.center,
              transformHitTests: false,
              child: widget.ratingWidget!.half,
            )
          : widget.ratingWidget!.half,
      ),
    );
  }
  iconRating += 0.5;
} else {
  ratingWidget = SizedBox(
    width: widget.itemSize,
    height: widget.itemSize,
    child: FittedBox(
      fit: BoxFit.contain,
      child: widget.ratingWidget?.full ?? widget.itemBuilder!(context, index),
    ),
}

```

```

    );
    iconRating += 1.0;
}

return IgnorePointer(
  ignoring: widget.ignoreGestures,
  child: GestureDetector(
    onTap: () {
      widget.onRatingUpdate(index + 1.0);
      setState(() {
        _rating = index + 1.0;
      });
    },
    onHorizontalDragStart: _isHorizontal ? (_) => _glow.value = true : null,
    onHorizontalDragEnd: _isHorizontal
      ? (_)
        ? _glow.value = false;
        widget.onRatingUpdate(iconRating);
        iconRating = 0.0;
      }
      : null,
    onHorizontalDragUpdate: _isHorizontal ? (dragUpdates) =>
_dragOperation(dragUpdates, widget.direction) : null,
    onVerticalDragStart: _isHorizontal ? null : (_) => _glow.value = true,
    onVerticalDragEnd: _isHorizontal
      ? null
      : (_)
        ? _glow.value = false;
        widget.onRatingUpdate(iconRating);
        iconRating = 0.0;
      },
    onVerticalDragUpdate: _isHorizontal ? null : (dragUpdates) =>
_dragOperation(dragUpdates, widget.direction),
    child: Padding(
      padding: widget.itemPadding,
      child: ValueListenableBuilder(
        valueListenable: _glow,
        builder: (context, dynamic glow, _) {
          if (glow && widget.glow) {
            Color glowColor = widget.glowColor ??
Theme.of(context).colorScheme.secondary;
            return DecoratedBox(
              decoration: BoxDecoration(
                shape: BoxShape.circle,
                boxShadow: [

```

```

        BoxShadow(
            color: glowColor.withOpacity(30),
            blurRadius: 10,
            spreadRadius: widget.glowRadius,
        ),
        BoxShadow(
            color: glowColor.withOpacity(20),
            blurRadius: 10,
            spreadRadius: widget.glowRadius,
        ),
    ],
),
child: ratingWidget,
);
} else {
    return ratingWidget;
}
},
),
),
),
);
}
},
),
),
),
);
}
bool get _isHorizontal => widget.direction == Axis.horizontal;

void _dragOperation(DragUpdateDetails dragDetails, Axis direction) {
if (!widget.tapOnlyMode) {
    RenderBox box = context.findRenderObject() as RenderBox;
    var pos = box.globalToLocal(dragDetails.globalPosition);
    double i;
    if (direction == Axis.horizontal) {
        i = pos.dx / (widget.itemSize + widget.itemPadding.horizontal);
    } else {
        i = pos.dy / (widget.itemSize + widget.itemPadding.vertical);
    }
    var currentRating = widget.allowHalfRating ? i : i.round().toDouble();
    if (currentRating > widget.itemCount) {
        currentRating = widget.itemCount.toDouble();
    }
    if (currentRating < 0) {
        currentRating = 0.0;
    }
    if (_isRTL && widget.direction == Axis.horizontal) {
        currentRating = widget.itemCount - currentRating;
    }
}
}

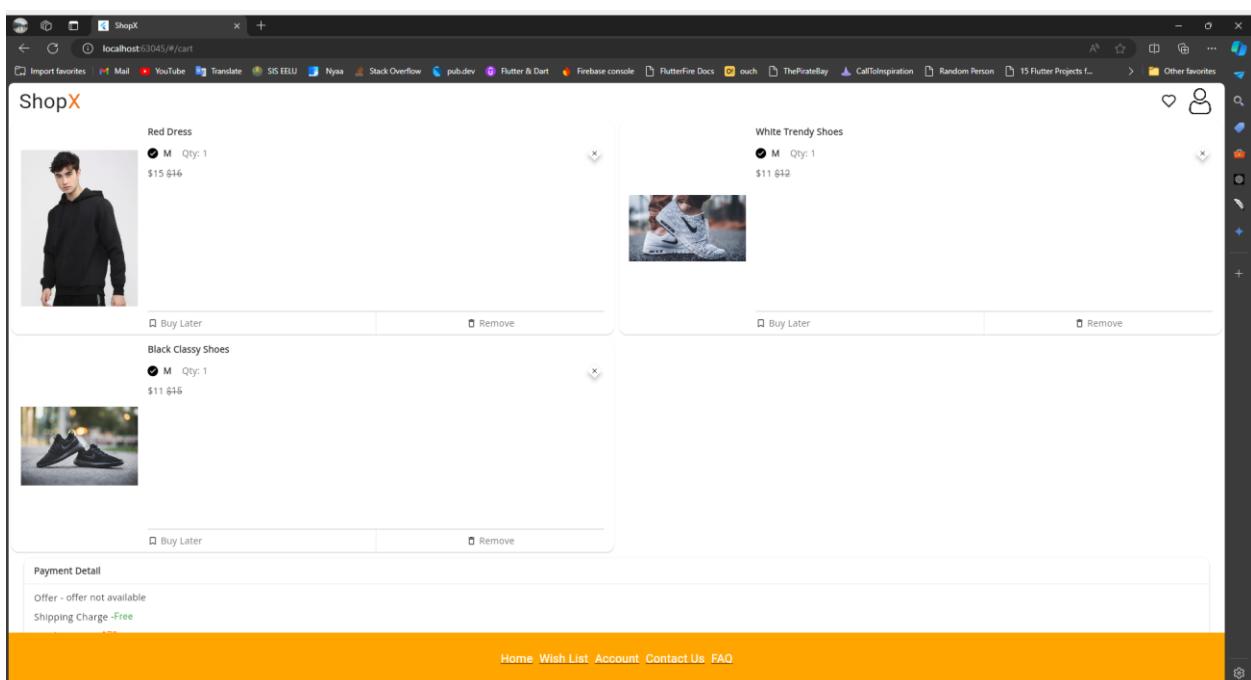
```

```

        }
        if (currentRating < _minRating!) {
            _rating = _minRating;
        } else if (currentRating > _maxRating!) {
            _rating = _maxRating;
        } else {
            _rating = currentRating;
        }
        setState(() {});
    }
}
}
}

```

## 5.2.4 Cart



## Code Implementation:

```

import 'package:ecommerce_responsive/utils/widgets/appbar.dart';
import 'package:ecommerce_responsive/utils/widgets/cart_item.dart';
import 'package:ecommerce_responsive/utils/widgets/footer.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart' hide ContextExtensionss;
import 'package:nb_utils/nb_utils.dart';

```

```

import 'package:ecommerce_responsive/models/model_product.dart';
import 'package:ecommerce_responsive/screens/screen_order_summary.dart';
import 'package:ecommerce_responsive/utils/colors_constant.dart';
import 'package:ecommerce_responsive/utils/size_constant.dart';
import 'package:ecommerce_responsive/utils/root_bundle.dart';
import 'package:ecommerce_responsive/utils/strings_constant.dart';
import 'package:ecommerce_responsive/utils/widgets/app_widget.dart';

class ScreenCart extends StatefulWidget {
    static const String tag = '/cart';

    const ScreenCart({super.key});

    @override
    ScreenCartState createState() => ScreenCartState();
}

class ScreenCartState extends State<ScreenCart> {
    List<ModelProduct> cartList = [];

    @override
    void initState() {
        super.initState();
        fetchData();
    }

    fetchData() async {
        var products = await loadCartProducts();
        setState(() {
            cartList.clear();
            cartList.addAll(products);
        });
    }

    static Widget paymentDetail =
    Container(
        margin: const EdgeInsets.symmetric(horizontal: 20),
        child: Card(
            elevation: 3,
            child: Column(
                mainAxisAlignment: MainAxisAlignment.start,
                children: [
                    Padding(
                        padding: const EdgeInsets.fromLTRB(spacing_standard_new,
spacing_middle, spacing_standard_new, spacing_middle),

```

```

        child: Text(sh_lbl_payment_details, style: boldTextStyle(size: 14)),
    ),
    const Divider(height: 1, color: sh_view_color),
    Padding(
        padding: const EdgeInsets.fromLTRB(spacing_standard_new,
spacing_middle, spacing_standard_new, spacing_middle),
        child: Column(
            children: [
                Row(
                    children: [
                        text(sh_lbl_offer,fontSize: 14),
                        4.width,
                        Text(sh_text_offer_not_available, style:
primaryTextStyle(size: 14)),
                    ],
                ),
                8.height,
                Row(
                    children: [
                        text(sh_lbl_shipping_charge,fontSize: 14),
                        text(sh_lbl_free, textColor: Colors.green, fontFamily:
fontMedium,fontSize: 14),
                    ],
                ),
                8.height,
                Row(
                    children: [
                        text(sh_lbl_total_amount,fontSize: 14),
                        text("\$70", textColor: sh_colorPrimary, fontFamily:
fontBold, fontSize: 14),
                    ],
                ),
                8.height,
                Row(
                    children: [
                        text(sh_lbl_subtotal,fontSize: 14),
                        text("100", textColor: sh_colorPrimary, fontFamily:
fontBold, fontSize: 14),
                    ],
                ),
                8.height,
                Row(
                    children: [
                        text(sh_lbl_discount,fontSize: 14),
                        text("0", textColor: sh_colorPrimary, fontFamily:
fontBold, fontSize: 14),
                    ],
                ),
                8.height,
                Row(
                    children: [
                        text(sh_lbl_grand_total,fontSize: 14),
                        text("100", textColor: sh_colorPrimary, fontFamily:
fontBold, fontSize: 14),
                    ],
                ),
            ],
        );
    );

    @override
    Widget build(BuildContext context) {
        var width = MediaQuery.of(context).size.width;
        var height = MediaQuery.of(context).size.height;
        var cartListWidget = context.isDesktop()?

```

```

    GridView.builder(
        gridDelegate:
    SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 2, childAspectRatio:
width<840? 2: 2.8),
        itemCount: cartList.length,
        shrinkWrap: true,
        itemBuilder: (context, index) {
            return IntrinsicHeight(child: CartItem(item:cartList[index]));
        })
: ListView.builder(
    itemCount: cartList.length,
    shrinkWrap: true,
    padding: const EdgeInsets.only(bottom: spacing_standard_new),
    physics: const NeverScrollableScrollPhysics(),
    itemBuilder: (context, index) {
        return CartItem(item:cartList[index]);
    });
}

var bottomButtons = Container(
    height: 65,
    decoration: const BoxDecoration(boxShadow: [
        BoxShadow(color: sh_shadow_color, blurRadius: 10, spreadRadius: 2,
offset: Offset(0, 3)),
    ], color: sh_white),
    child: Row(
        mainAxisAlignment: CrossAxisAlignmentAlignment.center,
        children: [
            Container(
                color: context.cardColor,
                child: Column(
                    mainAxisAlignment: CrossAxisAlignmentAlignment.center,
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                        Text("\$70", style: boldTextStyle()),
                        Text(sh_lbl_see_price_detail, style: secondaryTextStyle()),
                    ],
                ),
            ).expand(),
            Container(
                color: sh_colorPrimary,
                alignment: Alignment.center,
                height: double.infinity,
                child: text(sh_lbl_continue, textColor: sh_white, fontSize:
textSizeLargeMedium, fontFamily: fontMedium),

```

```

        ).onTap(()=> Get.toNamed(ScreenOrderSummary.tag),
        ).expand()
    ],
),
);
return Scaffold(
    appBar: const DefaultAppBar(title:"Cart"),
    body: SizedBox(
        height: height,
        child: Stack(
            children: [
                SingleChildScrollView(
                    child: Column(
                        children: [
                            cartListWidget,
                            paymentDetail,
                            50.height,
                            bottomButtons,
                            80.height,
                        ],
                    ),
                ),
                const Positioned.fill(
                    child: Align(
                        alignment: Alignment.bottomCenter,
                        child: Footer()
                    )
                ),
            ],
        ),
    ),
);
}
}

import 'package:ecommerce_responsive/main.dart';
import 'package:ecommerce_responsive/utils/colors_constant.dart';
import 'package:ecommerce_responsive/utils/images_constant.dart';
import 'package:ecommerce_responsive/utils/extension/currency_extension.dart';
import 'package:ecommerce_responsive/utils/size_constant.dart';
import 'package:ecommerce_responsive/utils/strings_constant.dart';
import 'package:ecommerce_responsive/utils/widgets/image_holder.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:nb_utils/nb_utils.dart';

```

```
import '../../models/model_product.dart';

class CartItem extends StatelessWidget {
  const CartItem({Key? key, required this.item}) : super(key: key);
  final ModelProduct item;

  @override
  Widget build(BuildContext context) {
    return Card(
      child: Container(
        height: context.isDesktop() || kIsWeb?145: context.height() * 0.25,
        margin: const EdgeInsets.only(left: spacing_standard_new, right:
spacing_standard_new),
        child: Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: [
            ImageHolder(imagePath:
"${base}img/products${item.images![0].src!}",),
            15.width,
            Expanded(
              child: Column(
                mainAxisSize: MainAxisSize.max,
                mainAxisAlignment: MainAxisAlignment.start,
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  5.height,
                  Text(item.name.toString(), style: boldTextStyle(size: 14)),
                  4.height,
                  Row(
                    children: [
                      Container(
                        decoration: const BoxDecoration(shape: BoxShape.circle,
color: Colors.black),
                        padding: const EdgeInsets.all(spacing_control_half),
                        child: const Icon(Icons.done, color: sh_white, size: 12),
                      ),
                      8.width,
                      Text("M", style: boldTextStyle(size: 14)),
                      8.width,
                      Text("Qty: 1", style:
secondaryTextStyle()).paddingOnly(left: 8),
                      const Spacer(),
                    ],
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

```

                shape: BeveledRectangleBorder(borderRadius:
BorderRadius.circular(20.0),),
                    elevation: 5,
                    child: const Icon(Icons.close, color: black, size:
12).paddingAll(5),
                        ),
                    ],
                ).paddingOnly(top: spacing_control),
6.height,
Row(
    mainAxisAlignment: MainAxisAlignment.end,
    children: [
        Text(
            item.on_sale! ?
item.sale_price.toString().toCurrencyFormat().toString() :
item.price.toString().toCurrencyFormat().toString(),
                style: primaryTextStyle(size: 14),
            ),
        4.width,
        Text(item.regular_price.toString().toCurrencyFormat()!,
            style: const TextStyle(color: sh_textColorSecondary,
fontFamily: fontRegular, fontSize: textSizeSMedium, decoration:
TextDecoration.lineThrough),
                ),
        ],
    ),
),

const Spacer(),
const Divider(height: 1,),
Row(
    children: [
        Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
                Icon(Icons.bookmark_border, color:
appStore.isDarkModeOn ? gray : sh_textColorPrimary, size: 16,),
                4.width,
                (context.width()> 716 && context.width()<780)?
const SizedBox.shrink():
Text("Buy Later", style: secondaryTextStyle()),
overflow: TextOverflow.ellipsis, maxLines: 1,.expand()
                ],
            ).expand(),
            Container(width: 1, color: sh_view_color, height: 35),
            Row(

```

```

        mainAxisAlignment: MainAxisAlignment.center,
        children: [
            Icon(Icons.delete_outline, color: appStore.isDarkModeOn
? gray : sh_textColorPrimary, size: 16,),
            4.width,
            (context.width()> 716 && context.width()<780)?
            const SizedBox.shrink():
            Text(sh_lbl_remove, style: secondaryTextStyle()),
            ],
        ).expand()
    ],
),
),
],
),
),
),
),
);
}
}

import 'package:intl/intl.dart';

extension StringExtension on String? {
String? toCurrencyFormat({var format = '\$'}) {
    return format + this;
}

String formatDateTime() {
    if (this == null || this!.isEmpty || this == "null") {
        return "NA";
    } else {
        return DateFormat("HH:mm dd MMM yyyy", "en_US").format(DateFormat("yyyy-MM-
dd HH:mm:ss.0", "en_US").parse(this!));
    }
}

String formatDate() {
    if (this == null || this!.isEmpty || this == "null") {
        return "NA";
    } else {
        return DateFormat("dd MMM yyyy", "en_US").format(DateFormat("yyyy-MM-dd",
"en_US").parse(this!));
    }
}

```

```
    }  
}
```

## 5.2.5 Wishlist

```
import 'dart:convert';  
  
import 'package:ecommerce_responsive/utils/images_constant.dart';  
import 'package:ecommerce_responsive/utils/widgets/appbar.dart';  
import 'package:ecommerce_responsive/utils/widgets/footer.dart';  
import 'package:ecommerce_responsive/utils/widgets/image_holder.dart';  
import 'package:flutter/material.dart';  
import 'package:nb_utils/nb_utils.dart';  
import 'package:ecommerce_responsive/models/model_product.dart';  
import 'package:ecommerce_responsive/utils/colors_constant.dart';  
import 'package:ecommerce_responsive/utils/size_constant.dart';  
import 'package:ecommerce_responsive/utils/root_bundle.dart';  
import 'package:ecommerce_responsive/utils/strings_constant.dart';  
import 'package:ecommerce_responsive/main.dart';  
import 'package:ecommerce_responsive/utils/extension/currency_extension.dart';  
  
class ScreenWishlist extends StatefulWidget {  
    static const String tag = '/wishList';  
  
    const ScreenWishlist({super.key});  
  
    @override  
    ScreenWishlistState createState() => ScreenWishlistState();  
}  
  
class ScreenWishlistState extends State<ScreenWishlist> {  
    List<ModelProduct> list = [];  
  
    @override  
    void initState() {  
        super.initState();  
        fetchData();  
    }  
  
    fetchData() async {  
        var products = await loadProducts();  
        setState(() {  
            list.clear();  
            list.addAll(products);  
        });  
    }  
}
```

```

    });
}

Future<List<ModelProduct>> loadProducts() async {
    String jsonString = await
loadContentAsset('assets/shopshop_data/wishlist_products.json');
    final jsonResponse = json.decode(jsonString);
    return (jsonResponse as List).map((i) => ModelProduct.fromJson(i)).toList();
}

@Override
void setState(fn) {
    if (mounted) super.setState(fn);
}

@Override
Widget build(BuildContext context) {

    return Scaffold(
        appBar: const DefaultAppBar(title: "Wishlist"),
        body: Column(
            children: [
                Expanded(child: context.isDesktop()?
                    GridView.builder(
                        gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
                            crossAxisCount: 2,
                            childAspectRatio:context.width()>1035? 4.5:3),
                        itemCount: list.length,
                        itemBuilder:(context, index) {
                            return wishListItem(list[index]);
                        }
                    ):ListView.builder(
                        scrollDirection: Axis.vertical,
                        itemCount: list.length,
                        shrinkWrap: true,
                        padding: const EdgeInsets.only(bottom: 70),
                        itemBuilder: (context, index) {
                            return wishListItem(list[index]);
                        },
                    ),),
                15.height,
                const Footer(),
            ],
        ),
    );
}

```

```

    );
}

Widget wishListItem(ModelProduct item){
    return Card(
        child: Container(
            color: appStore.isDarkModeOn ? scaffoldDarkColor : white,
            constraints: const BoxConstraints(maxHeight: 400),
            height: context.height() * 0.25,
            margin: const EdgeInsets.only(left: spacing_standard_new, right:
spacing_standard_new),
            child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: <Widget>[
                    ImageHolder(imagePath:
"${base}img/products${item.images![0].src!}" ),
                    10.width,
                    Column(
                        mainAxisAlignment: MainAxisAlignment.start,
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: <Widget>[
                            //8.height,
                            Text(item.name!, style: boldTextStyle()).paddingOnly(left: 16.0),
                            Text(item.regular_price.toString().toCurrencyFormat().toString(),
style: boldTextStyle(color: sh_colorPrimary)).paddingOnly(left: 16),
                            const Spacer(),
                            Wrap(
                                children: <Widget>[
                                    Row(
                                        mainAxisSize: MainAxisSize.min,
                                        children: <Widget>[
                                            Icon(Icons.add_shopping_cart, color:
sh_textColorPrimary, size: context.width()*0.03),
                                            (context.width() > 715 && context.width()<775)?
                                                const SizedBox.shrink():
                                                Text(sh_lbl_move_to_cart, style: primaryTextStyle(size:
context.isPhone()? (context.width()*0.03).toInt() :null )),
                                            ],
                                        ),
                                    15.width,
                                    Row(
                                        mainAxisSize: MainAxisSize.min,
                                        children: [

```

```

        Icon(Icons.delete_outline, color: sh_textColorPrimary,
size: context.width()*0.03),
        (context.width() > 710 && context.width()<910)?
        const SizedBox.shrink():
        Text(sh_lbl_remove, style: primaryTextStyle(size:
context.isPhone()? (context.width()*0.03).toInt() :null )),
        ],
        )
        ],
        ).expand()
        ],
        ).expand()
        ],
        ),
        ),
        );
    }
}

import 'package:flutter/material.dart';
import 'package:get/get_utils/get_utils.dart';

class ImageHolder extends StatelessWidget {
    const ImageHolder({Key? key, required this.imagePath}) : super(key: key);
    final String imagePath;
    @override
    Widget build(BuildContext context) {
        return ConstrainedBox(
            constraints: const BoxConstraints(maxWidth: 180, minWidth: 90),
            child: Image.asset(
                imagePath,
                width: context.width * 0.25,
                height: context.width * 0.3,
                fit: BoxFit.contain,
            ),
        );
    }
}

import 'model_category.dart';

class ModelProduct {
    int? id;
    String? name;

```

```
// ignore: non_constant_identifier_names
String? date_created;

// ignore: non_constant_identifier_names
String? date_created_gmt;

// ignore: non_constant_identifier_names
String? date_modified;

// ignore: non_constant_identifier_names
String? date_modified_gmt;
String? type;
String? status;
bool? featured;

// ignore: non_constant_identifier_names
String? catalog_visibility;
String? description;

// ignore: non_constant_identifier_names
String? short_description;
String? sku;
String? price;

// ignore: non_constant_identifier_names
String? regular_price;

// ignore: non_constant_identifier_names
String? sale_price;

// ignore: non_constant_identifier_names
String? price_html;

// ignore: non_constant_identifier_names
bool? on_sale;
bool? purchasable;

// ignore: non_constant_identifier_names
int? total_sales;
bool? virtual;
bool? downloadable;

// ignore: non_constant_identifier_names
```

```
String? external_url;

// ignore: non_constant_identifier_names
String? button_text;

// ignore: non_constant_identifier_names
String? tax_status;

// ignore: non_constant_identifier_names
String? tax_class;

// ignore: non_constant_identifier_names
bool? manage_stock;

// ignore: non_constant_identifier_names
int? stock_quantity;

// ignore: non_constant_identifier_names
String? stock_status;
String? backorders;

// ignore: non_constant_identifier_names
bool? backorders_allowed;
bool? backordered;

// ignore: non_constant_identifier_names
bool? sold_individually;
String? weight;
Dimensions? dimensions;

// ignore: non_constant_identifier_names
bool? shipping_required;

// ignore: non_constant_identifier_names
bool? shipping_taxable;

// ignore: non_constant_identifier_names
String? shipping_class;

// ignore: non_constant_identifier_names
int? shipping_class_id;

// ignore: non_constant_identifier_names
bool? reviews_allowed;
```

```
// ignore: non_constant_identifier_names
String? average_rating;

// ignore: non_constant_identifier_names
int? rating_count;

// ignore: non_constant_identifier_names
List<dynamic>? related_ids;

// ignore: non_constant_identifier_names
List<dynamic>? upsell_ids;

// ignore: non_constant_identifier_names
List<dynamic>? cross_sell_ids;

// ignore: non_constant_identifier_names
int? parent_id;

// ignore: non_constant_identifier_names
String? purchase_note;
List<ModelCategory>? categories;
List<dynamic>? tags;
List<ShImage>? images;
List<Attribute>? attributes;

// ignore: non_constant_identifier_names
List<dynamic>? default_attributes;

ModelProduct(
    {this.id,
    this.name,
    // ignore: non_constant_identifier_names
    this.date_created,
    // ignore: non_constant_identifier_names
    this.date_created_gmt,
    // ignore: non_constant_identifier_names
    this.date_modified,
    // ignore: non_constant_identifier_names
    this.date_modified_gmt,
    this.type,
    this.status,
    this.featured,
    // ignore: non_constant_identifier_names
    this.catalog_visibility,
    this.description,
```

```
// ignore: non_constant_identifier_names
this.short_description,
this.sku,
this.price,
// ignore: non_constant_identifier_names
this.regular_price,
// ignore: non_constant_identifier_names
this.sale_price,
// ignore: non_constant_identifier_names
this.price_html,
// ignore: non_constant_identifier_names
this.on_sale,
this.purchasable,
// ignore: non_constant_identifier_names
this.total_sales,
this.virtual,
this.downloadable,

// ignore: non_constant_identifier_names
this.external_url,
// ignore: non_constant_identifier_names
this.button_text,
// ignore: non_constant_identifier_names
this.tax_status,
// ignore: non_constant_identifier_names
this.tax_class,
// ignore: non_constant_identifier_names
this.manage_stock,
// ignore: non_constant_identifier_names
this.stock_quantity,
// ignore: non_constant_identifier_names
this.stock_status,
this.backorders,
// ignore: non_constant_identifier_names
this.backorders_allowed,
this.backordered,
// ignore: non_constant_identifier_names
this.sold_individually,
this.weight,
this.dimensions,
// ignore: non_constant_identifier_names
this.shipping_required,
// ignore: non_constant_identifier_names
this.shipping_taxable,
// ignore: non_constant_identifier_names
```

```

    this.shipping_class,
    // ignore: non_constant_identifier_names
    this.shipping_class_id,
    // ignore: non_constant_identifier_names
    this.reviews_allowed,
    // ignore: non_constant_identifier_names
    this.average_rating,
    // ignore: non_constant_identifier_names
    this.rating_count,
    // ignore: non_constant_identifier_names
    this.related_ids,
    // ignore: non_constant_identifier_names
    this.upsell_ids,
    // ignore: non_constant_identifier_names
    this.cross_sell_ids,
    // ignore: non_constant_identifier_names
    this.parent_id,
    // ignore: non_constant_identifier_names
    this.purchase_note,
    this.categories,
    this.tags,
    this.images,
    this.attributes,
    // ignore: non_constant_identifier_names
    this.default_attributes);
}

factory ModelProduct.fromJson(Map<String, dynamic> json) {
  return ModelProduct(
    id: json['id'],
    name: json['name'],
    date_created: json['date_created'],
    date_created_gmt: json['date_created_gmt'],
    date_modified: json['date_modified'],
    date_modified_gmt: json['date_modified_gmt'],
    type: json['type'],
    status: json['status'],
    featured: json['featured'],
    catalog_visibility: json['catalog_visibility'],
    description: json['description'],
    short_description: json['short_description'],
    sku: json['sku'],
    price: json['price'],
    regular_price: json['regular_price'],
    sale_price: json['sale_price'],
    price_html: json['price_html'],
  );
}

```

```

on_sale: json['on_sale'],
purchasable: json['purchasable'],
total_sales: json['total_sales'],
virtual: json['virtual'],
downloadable: json['downloadable'],
external_url: json['external_url'],
button_text: json['button_text'],
tax_status: json['tax_status'],
tax_class: json['tax_class'],
manage_stock: json['manage_stock'],
stock_quantity: json['stock_quantity'],
stock_status: json['stock_status'],
backorders: json['backorders'],
backorders_allowed: json['backorders_allowed'],
backordered: json['backordered'],
sold_individually: json['sold_individually'],
weight: json['weight'],
dimensions: json['dimensions'] != null ?
Dimensions.fromJson(json['dimensions']) : null,
shipping_required: json['shipping_required'],
shipping_taxable: json['shipping_taxable'],
shipping_class: json['shipping_class'],
shipping_class_id: json['shipping_class_id'],
reviews_allowed: json['reviews_allowed'],
average_rating: json['average_rating'],
rating_count: json['rating_count'],
parent_id: json['parent_id'],
purchase_note: json['purchase_note'],
categories: json['categories'] != null ? (json['categories'] as
List).map((i) => ModelCategory.fromJson(i)).toList() : null,
images: json['images'] != null ? (json['images'] as List).map((i) =>
ShImage.fromJson(i)).toList() : null,
attributes: json['attributes'] != null ? (json['attributes'] as
List).map((i) => Attribute.fromJson(i)).toList() : null,
);
}

Map<String, dynamic> toJson() {
final Map<String, dynamic> data = <String, dynamic>{};
data['id'] = id;
data['name'] = name;
data['date_created'] = date_created;
data['date_created_gmt'] = date_created_gmt;
data['date_modified'] = date_modified;
data['date_modified_gmt'] = date_modified_gmt;
}

```

```

data['type'] = type;
data['status'] = status;
data['featured'] = featured;
data['catalog_visibility'] = catalog_visibility;
data['description'] = description;
data['short_description'] = short_description;
data['sku'] = sku;
data['price'] = price;
data['regular_price'] = regular_price;
data['sale_price'] = sale_price;
data['price_html'] = price_html;
data['on_sale'] = on_sale;
data['purchasable'] = purchasable;
data['total_sales'] = total_sales;
data['virtual'] = virtual;
data['downloadable'] = downloadable;
data['external_url'] = external_url;
data['button_text'] = button_text;
data['tax_status'] = tax_status;
data['tax_class'] = tax_class;
data['manage_stock'] = manage_stock;
data['stock_quantity'] = stock_quantity;
data['stock_status'] = stock_status;
data['backorders'] = backorders;
data['backorders_allowed'] = backorders_allowed;
data['backordered'] = backordered;
data['sold_individually'] = sold_individually;
data['weight'] = weight;
if (dimensions != null) {
    data['dimensions'] = dimensions!.toJson();
}
data['shipping_required'] = shipping_required;
data['shipping_taxable'] = shipping_taxable;
data['shipping_class'] = shipping_class;
data['shipping_class_id'] = shipping_class_id;
data['reviews_allowed'] = reviews_allowed;
data['average_rating'] = average_rating;
data['rating_count'] = rating_count;

data['parent_id'] = parent_id;
data['purchase_note'] = purchase_note;
if (categories != null) {
    data['categories'] = categories!.map((v) => v.toJson()).toList();
}
if (tags != null) {

```

```

        data['tags'] = tags!.map((v) => v.toJson()).toList();
    }
    if (images != null) {
        data['images'] = images!.map((v) => v.toJson()).toList();
    }
    if (attributes != null) {
        data['attributes'] = attributes!.map((v) => v.toJson()).toList();
    }
    if (default_attributes != null) {
        data['default_attributes'] =
            default_attributes!.map((v) => v.toJson()).toList();
    }
    return data;
}
}

class ShImage {
    int? id;

    // ignore: non_constant_identifier_names
    String? date_created;

    // ignore: non_constant_identifier_names
    String? date_created_gmt;

    // ignore: non_constant_identifier_names
    String? date_modified;

    // ignore: non_constant_identifier_names
    String? date_modified_gmt;
    String? src;
    String? name;
    String? alt;

    // ignore: non_constant_identifier_names
    ShImage({this.id, this.date_created, this.date_created_gmt, this.date_modified,
this.date_modified_gmt, this.src, this.name, this.alt});

    factory ShImage.fromJson(Map<String, dynamic> json) {
        return ShImage(
            id: json['id'],
            date_created: json['date_created'],
            date_created_gmt: json['date_created_gmt'],
            date_modified: json['date_modified'],

```

```

        date_modified_gmt: json['date_modified_gmt'],
        src: json['src'],
        name: json['name'],
        alt: json['alt'],
    );
}

Map<String, dynamic> toJson() {
    final Map<String, dynamic> data = <String, dynamic>{};
    data['id'] = id;
    data['date_created'] = date_created;
    data['date_created_gmt'] = date_created_gmt;
    data['date_modified'] = date_modified;
    data['date_modified_gmt'] = date_modified_gmt;
    data['src'] = src;
    data['name'] = name;
    data['alt'] = alt;
    return data;
}
}

class Attribute {
    int? id;
    String? name;
    int? position;
    bool? visible;
    bool? variation;
    List<String>? options;

    Attribute({this.id, this.name, this.position, this.visible, this.variation,
    this.options});

    factory Attribute.fromJson(Map<String, dynamic> json) {
        return Attribute(
            id: json['id'],
            name: json['name'],
            position: json['position'],
            visible: json['visible'],
            variation: json['variation'],
            options: json['options'] != null ? List<String>.from(json['options']) :
null,
        );
    }
}

Map<String, dynamic> toJson() {

```

```

final Map<String, dynamic> data = <String, dynamic>{};
data['id'] = id;
data['name'] = name;
data['position'] = position;
data['visible'] = visible;
data['variation'] = variation;
if (options != null) {
  data['options'] = options;
}
return data;
}

class Dimensions {
  String? length;
  String? width;
  String? height;

  Dimensions({this.length, this.width, this.height});

  factory Dimensions.fromJson(Map<String, dynamic> json) {
    return Dimensions(
      length: json['length'],
      width: json['width'],
      height: json['height'],
    );
  }

  Map<String, dynamic> toJson() {
    final Map<String, dynamic> data = <String, dynamic>{};
    data['length'] = length;
    data['width'] = width;
    data['height'] = height;
    return data;
  }
}

import 'dart:convert';

import 'package:flutter/services.dart';
import '../models/model_address.dart';
import '../models/model_attribute.dart';
import '../models/model_category.dart';
import '../models/model_order.dart';
import '../models/model_product.dart';

```

```

Future<String> loadContentAsset(String path) async {
    return await rootBundle.loadString(path);
}

Future<List<ModelCategory>> loadCategory() async {
    String jsonString = await
loadContentAsset('assets/shopshop_data/category.json');
    final jsonResponse = json.decode(jsonString);
    return (jsonResponse as List).map((i) => ModelCategory.fromJson(i)).toList();
}

Future<List<ModelProduct>> loadProducts() async {
    String jsonString = await
loadContentAsset('assets/shopshop_data/products.json');
    final jsonResponse = json.decode(jsonString);
    return (jsonResponse as List).map((i) => ModelProduct.fromJson(i)).toList();
}

Future<List<ModelProduct>> loadCartProducts() async {
    String jsonString = await
loadContentAsset('assets/shopshop_data/cart_products.json');
    final jsonResponse = json.decode(jsonString);
    return (jsonResponse as List).map((i) => ModelProduct.fromJson(i)).toList();
}

Future<ModelAttributes> loadAttributes() async {
    String jsonString = await
loadContentAsset('assets/shopshop_data/attributes.json');
    final jsonResponse = json.decode(jsonString);
    return ModelAttributes.fromJson(jsonResponse);
}

Future<List<ModelAddress>> loadAddresses() async {
    String jsonString = await loadContentAsset('assets/shopshop_data/address.json');
    final jsonResponse = json.decode(jsonString);
    return (jsonResponse as List).map((i) => ModelAddress.fromJson(i)).toList();
}

Future<List<ModelOrder>> loadOrders() async {
    String jsonString = await loadContentAsset('assets/shopshop_data/orders.json');
    final jsonResponse = json.decode(jsonString);
    return (jsonResponse as List).map((i) => ModelOrder.fromJson(i)).toList();
}

```

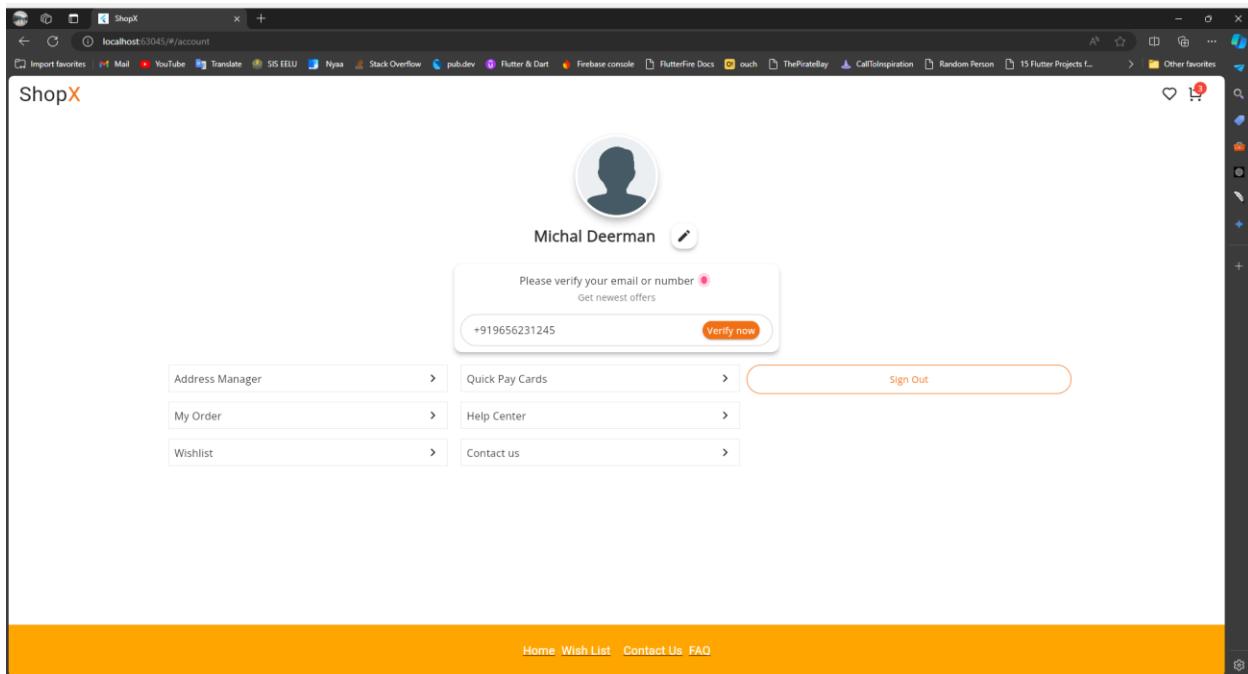
```

Future<List<String>> loadBanners() async {
  List<ModelProduct> products = await loadProducts();
  List<String> banner = [];

  products.forEach((product) {
    if (product.images!.isNotEmpty) {
      banner.add("assets/images/shopx/img/products${product.images![0].src!}");
    }
  });
  return banner;
}

```

## 5.2.6 Profile



## Code Implementation:

```

import 'package:ecommerce_responsive/utils/widgets/appbar.dart';
import 'package:ecommerce_responsive/utils/widgets/footer.dart';
import 'package:ecommerce_responsive/utils/widgets/material_button.dart';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:nb_utils/nb_utils.dart';
import 'package:ecommerce_responsive/utils/colors_constant.dart';

```

```
import 'package:ecommerce_responsive/utils/size_constant.dart';
import 'package:ecommerce_responsive/utils/images_constant.dart';
import 'package:ecommerce_responsive/utils/strings_constant.dart';

class ScreenProfile extends StatefulWidget {
    static const String tag = '/profile';

    const ScreenProfile({super.key});

    @override
    ScreenProfileState createState() => ScreenProfileState();
}

class ScreenProfileState extends State<ScreenProfile> {
    var emailCont = TextEditingController();
    var passwordCont = TextEditingController();
    var confirmPasswordCont = TextEditingController();
    var firstNameCont = TextEditingController();
    var lastNameCont = TextEditingController();
    String? selectedValue = "Male";

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: const DefaultAppBar(title: "Profile",),
            body: Column(
                children: [
                    Expanded(
                        child: SingleChildScrollView(
                            child: Padding(
                                padding: const EdgeInsets.all(spacing_standard_new),
                                child: Column(
                                    mainAxisAlignment: MainAxisAlignment.center,
                                    crossAxisAlignment: CrossAxisAlignment.center,
                                    children: <Widget>[
                                        const SizedBox(height: 10,),

                                        Wrap(
                                            alignment: WrapAlignment.center,
                                            children: [
                                                SizedBox(
                                                    width: context.isPhone()? double.maxFinite: 400,
                                                    child: Center(
                                                        child: Stack(
                                                            alignment: Alignment.bottomRight,
```

```

        children: <Widget>[
            Padding(
                padding: const
        EdgeInsets.all(spacing_standard_new),
                    child: Card(
                        semanticContainer: true,
                        clipBehavior: Clip.antiAliasWithSaveLayer,
                        elevation: spacing_standard,
                        margin: const
        EdgeInsets.all(spacing_control),
                    shape: RoundedRectangleBorder(borderRadius:
        BorderRadius.circular(100.0)),
                    child: const CircleAvatar(
                        backgroundImage: AssetImage(ic_user),
                        radius: 55,
                        ).paddingAll(4),
                    ),
                    ),
                    GestureDetector(
                        onTap: ()async {
                            await ImagePicker().pickMultiImage(
                                imageQuality: 100,
                                );
                        },
                        child: Container(
                            padding: const
        EdgeInsets.all(spacing_control),
                            margin: const EdgeInsets.only(bottom: 30,
right: 20),
                            decoration: BoxDecoration(
                                shape: BoxShape.circle,
                                color: context.cardColor,
                                border: Border.all(color: sh_colorPrimary,
width: 1),
                                ),
                            child: const Icon(Icons.camera_alt, color:
sh_colorPrimary, size: 16),
                            ),
                        ),
                    ],
                    ),
                ),
            ),
        ],
        SizedBox(

```

```

        width: context.isPhone() ? double.maxFinite : 600,
        child: Column(
            children: [
                Row(
                    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                    children: <Widget>[
                        Expanded(
                            child: TextFormField(
                                keyboardType: TextInputType.text,
                                autofocus: false,
                                controller: firstNameCont,
                                textCapitalization:
TextCapitalization.words,
                                style: primaryTextStyle(),
                                decoration: InputDecoration(
                                    filled: false,
                                    hintText: sh_hint_first_name,
                                    hintStyle: primaryTextStyle(),
                                    contentPadding: const
EdgeInsets.fromLTRB(
                            20.0, 15.0, 20.0, 15.0),
                                    focusedBorder: OutlineInputBorder(
                                        borderRadius:
BorderRadius.circular(32.0),
                                        borderSide: BorderSide(
                                            color:
Colors.grey.withOpacity(0.5),
                                            width: 0.5),
                                    ),
                                    enabledBorder: OutlineInputBorder(
                                        borderRadius:
BorderRadius.circular(32.0),
                                        borderSide: BorderSide(
                                            color:
Colors.grey.withOpacity(0.5),
                                            width: 0),
                                    ),
                                    ),
                                    ),
                                    ),
                                    ),
                                    const SizedBox(
                                        width: spacing_standard_new,
                                    ),
                                    Expanded(
                                        child: TextFormField(

```

```

        keyboardType: TextInputType.text,
        autofocus: false,
        controller: lastNameCont,
        textCapitalization:
TextCapitalization.words,
        style: primaryTextStyle(),
        decoration: InputDecoration(
            filled: false,
            hintText: sh_hint_last_name,
            hintStyle: primaryTextStyle(),
            contentPadding: const
EdgeInsets.fromLTRB(
                20.0, 15.0, 20.0, 15.0),
        focusedBorder: OutlineInputBorder(
            borderRadius:
BorderRadius.circular(32.0),
            borderSide: BorderSide(
                color:
Colors.grey.withOpacity(0.5),
                width: 0.5),
        ),
        enabledBorder: OutlineInputBorder(
            borderRadius:
BorderRadius.circular(32.0),
            borderSide: BorderSide(
                color:
Colors.grey.withOpacity(0.5),
                width: 0),
            ),
            ),
            ),
            ],
        ),
        const SizedBox(height: spacing_standard_new),
        Container(
            padding:
            const EdgeInsets.fromLTRB(20.0, 0.0, 20.0,
0.0),
            decoration: BoxDecoration(
                border: Border.all(
                    color: Colors.grey.withOpacity(0.3), width:
1),
                borderRadius:

```

```

                const
BorderRadius.all(Radius.circular(32.0)),
),
child: DropdownButton<String>(
underline: const SizedBox(),
isExpanded: true,
items: <String>["Male", "Female",
"Other"].map((String value) {
return DropdownMenuItem<String>(
value: value,
child: Text(value, style:
primaryTextStyle()),
);
}),
//hint:Text(selectedValue),
value: selectedValue,
onChanged: (newVal) {
setState(() {
selectedValue = newVal;
});
},
),
const SizedBox(height: spacing_standard_new),
 TextFormField(
keyboardType: TextInputType.emailAddress,
autofocus: false,
controller: emailCont,
textCapitalization: TextCapitalization.words,
style: primaryTextStyle(),
decoration: InputDecoration(
filled: false,
hintText: sh_hint_Email,
hintStyle: primaryTextStyle(),
contentPadding: const
EdgeInsets.fromLTRB(20.0, 15.0, 20.0, 15.0),
focusedBorder: OutlineInputBorder(
borderRadius: BorderRadius.circular(32.0),
borderSide: BorderSide(color:
Colors.grey.withOpacity(0.5), width: 0.5),
),
enabledBorder: OutlineInputBorder(
borderRadius: BorderRadius.circular(32.0),
borderSide: BorderSide(color:
Colors.grey.withOpacity(0.5), width: 0),),

```



```

}

import 'package:ecommerce_responsive/screens/screen_account.dart';
import 'package:ecommerce_responsive/screens/screen_contact_us.dart';
import 'package:ecommerce_responsive/screens/screen_faq.dart';
import 'package:ecommerce_responsive/screens/screen_home.dart';
import 'package:ecommerce_responsive/screens/screen_wishlist.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:nb_utils/nb_utils.dart';

import '../app_constant.dart';

class Footer extends StatelessWidget {
  const Footer({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      height: 80,
      color: orange,
      padding: const EdgeInsets.symmetric(horizontal: 20),
      child: Center(
        child: Wrap(
          children: [
            if(Get.routing.route!.settings.name != ScreenHome.tag)
              footerItem("Home", ScreenHome.tag),

            10.width,
            if(Get.routing.route!.settings.name != ScreenWishlist.tag)
              footerItem("Wish List", ScreenWishlist.tag),

            10.width,
            if(Get.routing.route!.settings.name != ScreenAccount.tag)
              footerItem("Account", ScreenAccount.tag),

            10.width,
            if(Get.routing.route!.settings.name != ScreenContactUs.tag)
              footerItem("Contact Us", ScreenContactUs.tag),

            10.width,
            if(Get.routing.route!.settings.name != ScreenFAQ.tag)
              footerItem("FAQ", ScreenFAQ.tag),
          ],
        ),
      ),
    );
  }
}

```

```

        ],
    ),
),
);
}
}

Widget footerItem(String item,String routeName){
    return Text(item,style: const TextStyle(fontSize:
textSizeLargeMedium,fontFamily: fontMedium,color: white,decoration:
TextDecoration.underline),).onTap((){
        Get.toNamed(routeName);
    });
}
}

import 'package:ecommerce_responsive/utils/colors_constant.dart';
import 'package:ecommerce_responsive/utils/size_constant.dart';
import 'package:ecommerce_responsive/utils/strings_constant.dart';
import 'package:ecommerce_responsive/utils/widgets/app_widget.dart';
import 'package:flutter/material.dart';

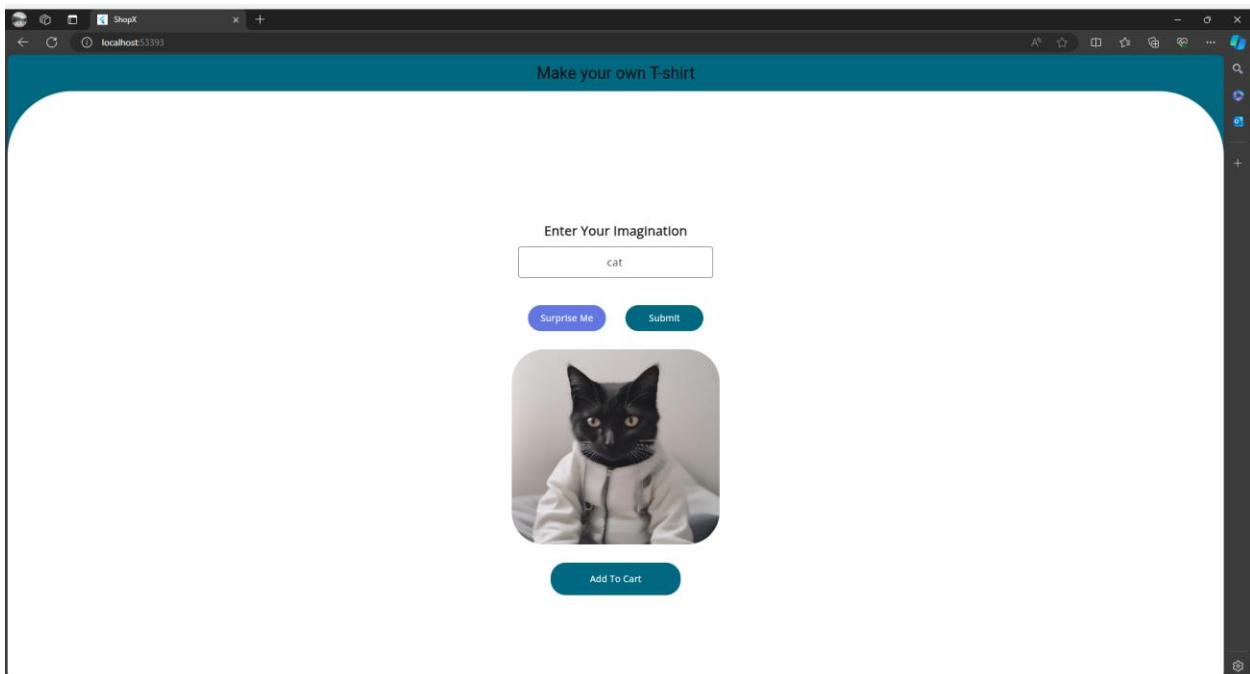
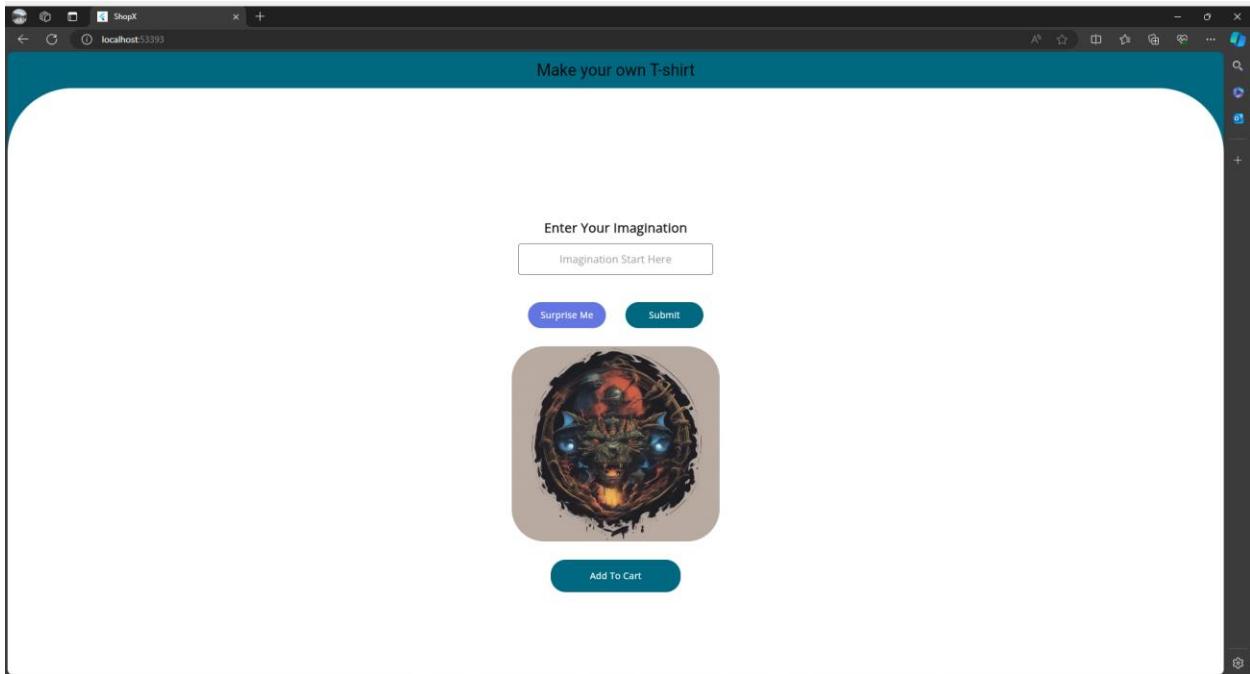
class RoundedButton extends StatelessWidget {
    const RoundedButton({Key? key, required this.onPress, this.title,
this.color,this.titleColor,this.width}) : super(key: key);
    final VoidCallback onPress;
    final String? title;
    final double ? width;
    final Color? color;
    final Color? titleColor;

    @override
    Widget build(BuildContext context) {
        return MaterialButton(
            textColor: sh_colorPrimary,
            color: color,
            minWidth: width,
            padding: const EdgeInsets.only(left: spacing_standard_new, right:
spacing_standard_new, top: 0, bottom: 0),
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(spacing_large),
                side: const BorderSide(color: sh_colorPrimary),
            ),
            onPressed: onPress,
            child: text(title??sh_lbl_add_card, fontSize: textSizeSMedium, textColor:
titleColor??sh_colorPrimary),
        );
    }
}

```

```
    );
}
}
```

## 5.2.7 generate images



## Code Implementation:

```
import 'dart:convert';
import 'dart:typed_data';

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:http/http.dart';

class GenerateTshirtApi {
  Future<Uint8List> post(
    required String url,
    required Map<String, dynamic> body,
    required String token) async {
    Map<String, String> headers = {
      'Authorization': 'Bearer $token',
      "accept": "image/*"
    };
    Response response = await http.post(Uri.parse(url),
        body: jsonEncode(body), headers: headers);
    if (response.statusCode == 200) {
      return response.bodyBytes;
    } else {
      throw Exception('Failed to load image');
    }
  }
}

class GenerateImage extends StatefulWidget {
  const GenerateImage({super.key});

  static const String id = 'GenerateT-shirt';

  @override
  // ignore: library_private_types_in_public_api
  _GenerateTshirtState createState() => _GenerateTshirtState();
}

class _GenerateTshirtState extends State<GenerateImage> {
  bool visible = false;
  String custimizeImagination = '';
  final String defaultImagination = 'TshirtDesignAF, T Shirt Design';
  Future<Uint8List>? _imageFuture;
  String? errorMessage;
```

```

@Override
void initState() {
    super.initState();
    _fetchImage(imagination: custimizeImagination);
}

void _fetchImage({required String imagination}) {
    setState(() {
        _imageFuture = GenerateTshirtApi().post(
            url:
                'https://api-
inference.huggingface.co/models/artificialguybr/TshirtDesignRedmond-V2',
            token: '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',
            body: {
                'inputs': imagination,
                'timestamp': DateTime.now().millisecondsSinceEpoch * .0000007
            },
        );
    });
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: const Text('Make your own T-shirt'),
            centerTitle: true,
            titleTextStyle: const TextStyle(fontSize: 25),
            elevation: 0,
            backgroundColor: kPrimaryColor,
        ),
        body: Background(
            child: SingleChildScrollView(
                child: Column(
                    children: [
                        const SizedBox(height: 150),
                        const Center(
                            child: Text(
                                'Enter Your Imagination',
                                style: TextStyle(
                                    fontSize: 20,
                                    fontWeight: FontWeight.w700,
                                ),
                            ),
                        ),
                    ],
                ),
            ),
        ),
    );
}

```

```
        const SizedBox(height: 10),
        SizedBox(
            width: 300,
            height: 90,
            child: TextField(
                onChanged: (value) {
                    setState(() {
                        custimizeImagination = value;
                        if (value.isEmpty) {
                            custimizeImagination = 'TshirtDesignAF, T Shirt Design';
                        }
                    });
                },
                enableSuggestions: true,
                textAlign: TextAlign.center,
                decoration: const InputDecoration(
                    border: OutlineInputBorder(),
                    hintText: 'Imagination Start Here',
                    hintStyle: TextStyle(color: Colors.grey),
                ),
            ),
        ),
    ),
    Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
            Button(
                textSize: 14,
                text: 'Surprise Me',
                tap: () {
                    setState(() {
                        _fetchImage(imagination: defaultImagination);
                        visible = true;
                    });
                },
                width: 120,
                colorBtn: const Color(0xCC3D54DA),
                colorTxt: kSecondaryColor,
                height: 40,
            ),
            const SizedBox(width: 30),
            Button(
                textSize: 14,
                text: 'Submit',
                tap: () {
                    if (custimizeImagination != '') {

```

```

        _fetchImage(imagination: custimizeImagination);
        visible = true;
    }
},
width: 120,
colorBtn: kPrimaryColor,
colorTxt: kSecondaryColor,
height: 40,
),
],
),
const SizedBox(height: 20),
Visibility(
visible: visible,
child: FutureBuilder<Uint8List>(
future: _imageFuture,
builder: (context, snapshot) {
if (snapshot.connectionState == ConnectionState.waiting) {
return const CircularProgressIndicator();
} else if (snapshot.hasError || errorMessage != null) {
return Center(
child: Text(
errorMessage ?? 'Failed to load image',
style: const TextStyle(color: Colors.red),
),
);
} else {
return GenerateTshirtResult(bytes: snapshot.data!);
}
},
),
),
],
),
),
);
}
}

class GenerateTshirtResult extends StatelessWidget {
const GenerateTshirtResult({super.key, required this.bytes});
final Uint8List bytes;
@Override
Widget build(BuildContext context) {

```

```

return Column(
  children: [
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Container(
        decoration: const BoxDecoration(
          borderRadius: BorderRadius.all(Radius.circular(50)),
        ),
        width: 320,
        height: 300,
        clipBehavior: Clip.antiAlias,
        child: Image.memory(
          bytes,
          fit: BoxFit.fill,
        ),
      ),
    ),
    Padding(
      padding: const EdgeInsets.all(20),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Button(
            textSize: 14,
            text: 'Add To Cart',
            tap: () {},
            width: 200,
            colorBtn: kPrimaryColor,
            colorTxt: kSecondaryColor,
            height: 50,
          ),
        ],
      ),
    ),
  ],
);
}

const Color kPrimaryColor = Color(0xFF006880);
const Color kSecondaryColor = Colors.white;

Map<String, WidgetBuilder> get routes {
  return {
    GenerateImage.id: (context) => const GenerateImage(),
  }
}

```

```
};

}

// This class defines a Background widget that wraps its child with a specific
design and color scheme.

class Background extends StatelessWidget {
    const Background(
        {super.key,
        required this.child}); // Constructor requires a child widget to be
provided.

    final Widget child; // The child widget to be wrapped by the Background.

    @override
    Widget build(BuildContext context) {
        // Builds the Background widget with a nested structure for design.
        return Container(
            color:
                kPrimaryColor, // Sets the background color to a predefined primary
color.
            width: double.infinity, // Expands the width to fill the available space.
            height:
                double.infinity, // Expands the height to fill the available space.
            child: Container(
                decoration: const BoxDecoration(
                    // Applies a decoration to create a curved effect after the AppBar.
                    borderRadius: BorderRadius.only(
                        topLeft: Radius.circular(100), topRight: Radius.circular(100)),
                    color: kSecondaryColor, // Sets the color of the inner container.
                ),
                child: Padding(
                    padding: const EdgeInsets.symmetric(
                        horizontal: 50,
                        vertical: 50),
                ), // Applies padding based on screen width.
                child: Container(
                    decoration: const BoxDecoration(
                        borderRadius: BorderRadius.all(
                            Radius.circular(
                                20), // Rounded corners for the innermost container.
                        ),
                    ),
                    width: 200, // Sets width relative to screen size.
                    height: 100, // Sets height relative to screen size.
                    child: child, // Places the provided child widget inside.
                ),
            ),
        );
    }
}
```

```
        ),
        ),
        );
    }
}

// This class defines a customizable Button widget that detects taps.

// ignore: must_be_immutable
class Button extends StatelessWidget {
    // Constructor requires parameters for text, tap action, dimensions, and
    // colors.
    Button(
        {super.key,
        required this.text, // Text to display on the button.
        required this.tap, // Function to execute on tap.
        required this.width, // Width of the button.
        required this.colorBtn, // Color of the button.
        required this.colorTxt, // Color of the text.
        required this.height, // Height of the button.
        this.textSize}); // Optional text size, defaults to 16 if not provided.

    // Fields for the button's customization parameters.
    String text;
    VoidCallback tap;
    double width;
    double height;
    Color colorBtn;
    Color colorTxt;
    double? textSize;

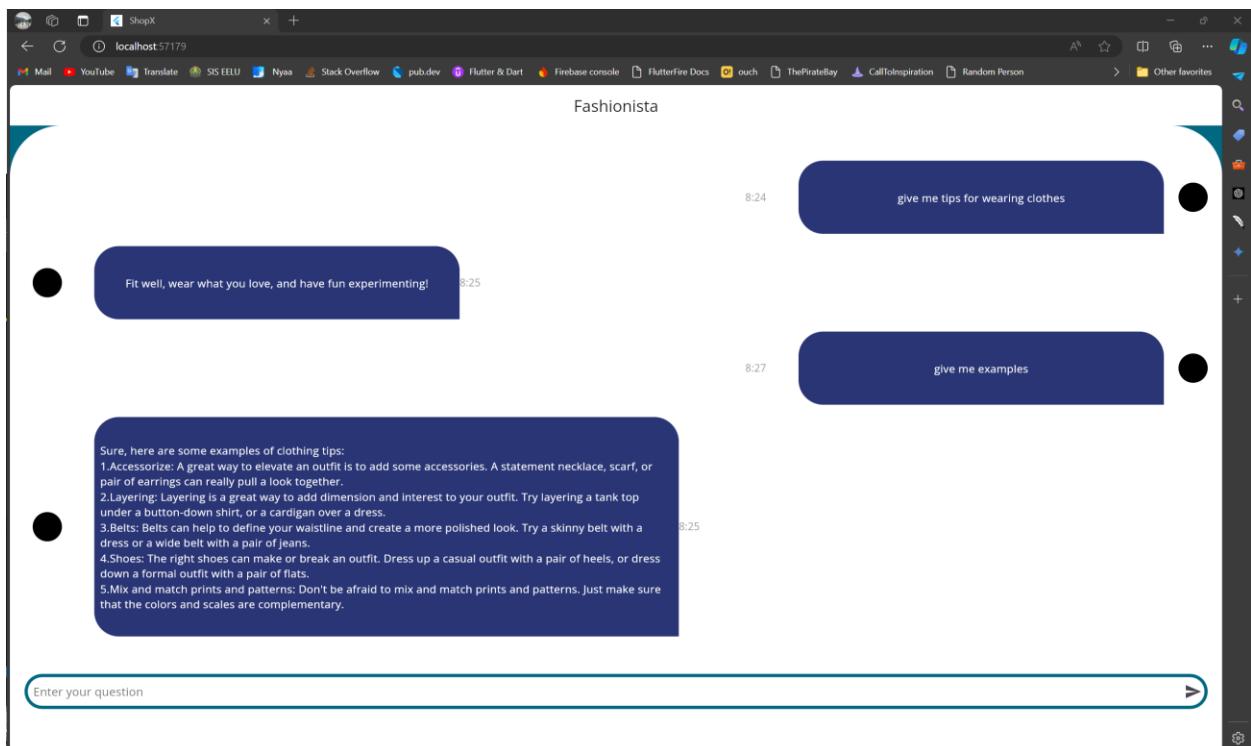
    @override
    Widget build(BuildContext context) {
        // Builds the Button widget with a GestureDetector to handle taps.
        return GestureDetector(
            onTap:
                tap, // Executes the provided tap function when the button is tapped.
            child: Container(
                decoration: BoxDecoration(
                    borderRadius: const BorderRadius.all(
                        Radius.circular(23)), // Rounded corners for the button.
                    color: colorBtn, // Background color of the button.
                ),
                width: width, // Sets the button's width.
                height: height, // Sets the button's height.
            ),
        );
    }
}
```

```

        child: Center(
            child: Text(
                text, // Displays the provided text.
                style: TextStyle(
                    fontSize: textSize ?? 16,
                    color: colorTxt), // Applies the text style.
            ),
        ),
    ),
);
}
}

```

## 5.2.8 ChatBot



## Code Implementation:

```

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:gap/gap.dart';

class ChatBotView extends StatelessWidget {
    const ChatBotView({Key? key}) : super(key: key);

```

```
static String id = 'chatbot';

@Override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Fashionista'),
      centerTitle: true,
    ),
    body: Background(
      child: Column(
        children: [
          Gap(ScreenUtil().setHeight(20)),
          Column(
            children: [
              Row(mainAxisAlignment: MainAxisAlignment.end, children: [
                Padding(
                  padding:
                    const EdgeInsets.only(left: 10, top: 5, bottom: 10).r,
                child: Align(
                  child: Row(
                    children: [
                      const Text(
                        '8:24',
                        style: TextStyle(color: Colors.grey),
                      ),
                      Gap(ScreenUtil().setWidth(10)),
                      Container(
                        height: 100,
                        width: 500,
                        decoration: BoxDecoration(
                          color: const Color(0xFF2a3575),
                          borderRadius: const BorderRadius.only(
                            topLeft: Radius.circular(30),
                            topRight: Radius.circular(30),
                            bottomRight: Radius.circular(0),
                            bottomLeft: Radius.circular(30),
                          ).r,
                        ),
                        child: const Center(
                          child: Text(
                            'give me tips for wearing clothes',
                            style: TextStyle(
                              color: Colors.white, fontSize: 15),
                          ),
                        ),
                      ),
                    ],
                  ),
                ),
              ],
            ],
          ),
        ],
      ),
    ),
  );
}
```

```
        ),
    ),
),
const Gap(20),
const CircleAvatar(
    backgroundColor: Colors.black,
),
],
),
),
),
),
),
// const Padding(
//   padding: EdgeInsets.only(right: 30),
//   child: Text(
//     '7:54',
//     style: TextStyle(color: Colors.grey),
//   ),
// ),
]),
Row(mainAxisAlignment: MainAxisAlignment.start, children: [
Padding(
padding:
    const EdgeInsets.only(left: 10, top: 5, bottom: 10).r,
child: Align(
child: Row(
children: [
const CircleAvatar(
    backgroundColor: Colors.black,
),
Gap(ScreenUtil().setWidth(10)),
Container(
height: 100,
width: 500,
decoration: BoxDecoration(
color: const Color(0xFF2a3575),
borderRadius: const BorderRadius.only(
    topLeft: Radius.circular(30),
    topRight: Radius.circular(30),
    bottomRight: Radius.circular(0),
    bottomLeft: Radius.circular(30),
).r,
),
child: const Center(
child: Text(
    'Fit well, wear what you love, and have fun
experimenting!',
```



```
        ),
        child: const Center(
            child: Text(
                'give me examples',
                style: TextStyle(
                    color: Colors.white, fontSize: 15),
            ),
        ),
    ),
),
const Gap(20),
const CircleAvatar(
    backgroundColor: Colors.black,
),
],
),
),
),
),
// const Padding(
//     padding: EdgeInsets.only(right: 30),
//     child: Text(
//         '7:54',
//         style: TextStyle(color: Colors.grey),
//     ),
// ),
// ],
),
Row(mainAxisAlignment: MainAxisAlignment.start, children: [
Padding(
padding:
    const EdgeInsets.only(left: 10, top: 5, bottom: 10).r,
child: Align(
    child: Row(
    children: [
        const CircleAvatar(
            backgroundColor: Colors.black,
        ),
        Gap(ScreenUtil().setWidth(10)),
        Container(
            height: 300,
            width: 800,
            decoration: BoxDecoration(
                color: const Color(0xFF2a3575),
                borderRadius: const BorderRadius.only(
                    topLeft: Radius.circular(30),
                    topRight: Radius.circular(30),
                    bottomRight: Radius.circular(0),

```

```
        bottomLeft: Radius.circular(30),
    ).r,
),
child: const Center(
    child: Padding(
        padding: EdgeInsets.all(8.0),
        child: Text(
            'Sure, here are some examples of clothing tips:
\n1.Accessorize: A great way to elevate an outfit is to add some accessories. A
statement necklace, scarf, or pair of earrings can really pull a look
together.\n2.Layering: Layering is a great way to add dimension and interest to
your outfit. Try layering a tank top under a button-down shirt, or a cardigan
over a dress.\n3.Belts: Belts can help to define your waistline and create a more
polished look. Try a skinny belt with a dress or a wide belt with a pair of
jeans.\n4.Shoes: The right shoes can make or break an outfit. Dress up a casual
outfit with a pair of heels, or dress down a formal outfit with a pair of
flats.\n5.Mix and match prints and patterns: Don\'t be afraid to mix and match
prints and patterns. Just make sure that the colors and scales are
complementary.',
```

```
style: TextStyle(  
    color: Colors.white, fontSize: 15),
```

),

），

9

```
const Text(
```

'8:25',

```
style: TextStyle(color: Colors.grey),
```

），

```
const Gap(20),
```

]  
,

,

),

```
// const Padding(
```

```
// padding: Edg
```

// child: Te

// '7:54',

// style:

10

**Gap(40),**

4

---

```

        TextFieldWidget(
            hintText: 'Enter your question',
            onSubmitted: (value) {},
            suffixIcon: IconButton(
                onPressed: () {},
                icon: const Icon(Icons.send),
            ),
        ),
    ],
),
),
),
),
);
}
}

// user_message.dart

class UserMessage extends StatelessWidget {
UserMessage(
    {Key? key,
    required this.date,
    required this.title,
    required this.isChatbot})
    : super(key: key);
String title;
String date;
bool isChatbot;
@Override
Widget build(BuildContext context) {
    return Row(mainAxisAlignment: MainAxisAlignment.end, children: [
    Padding(
        padding: const EdgeInsets.only(left: 10, top: 5, bottom: 10).r,
        child: Align(
            alignment: isChatbot ? Alignment.bottomLeft : Alignment.bottomRight,
            child: Row(
                children: [
                Text(
                    date,
                    style: const TextStyle(color: Colors.grey),
                ),
                Gap(ScreenUtil().setWidth(10)),
                Container(
                    height: 100,
                    width: 500,
                    decoration: BoxDecoration(

```

```

        color: const Color(0xFF2a3575),
        borderRadius: const BorderRadius.only(
            topLeft: Radius.circular(30),
            topRight: Radius.circular(30),
            bottomRight: Radius.circular(0),
            bottomLeft: Radius.circular(30),
        ).r,
    ),
    child: Center(
        child: Text(
            title,
            style: const TextStyle(color: Colors.white, fontSize: 15),
        ),
    ),
),
const Gap(20),
const CircleAvatar(
    backgroundColor: Colors.black,
),
],
),
),
),
),
// const Padding(
//     padding: EdgeInsets.only(right: 30),
//     child: Text(
//         '7:54',
//         style: TextStyle(color: Colors.grey),
//     ),
// ),
],
);
}
}

class Background extends StatelessWidget {
const Background(
    {super.key,
    required this.child,
    this.horizontalPadding,
    this.verticalPadding});

final Widget child;
final double? horizontalPadding;
final double? verticalPadding;

```

```

@Override
Widget build(BuildContext context) {
  return Container(
    color: kPrimaryColor,
    width: double.infinity,
    height: double.infinity,
    child: Container(
      decoration: const BoxDecoration(
        borderRadius: BorderRadius.only(
          topLeft: Radius.circular(70), topRight: Radius.circular(70)),
        color: Colors.white,
      ),
      child: Padding(
        padding: EdgeInsets.symmetric(
          horizontal: horizontalPadding ?? 20,
          vertical: verticalPadding ?? 20,
        ),
        child: Container(
          decoration: const BoxDecoration(
            borderRadius: BorderRadius.all(
              Radius.circular(20),
            ),
            ),
            child: child,
          ),
        ),
      ),
    );
}

const Color kPrimaryColor = Color(0xFF006880);

class TextFieldWidget extends StatelessWidget {
  const TextFieldWidget({
    super.key,
    this.hintText,
    this.tap,
    this.onSubmitted,
    this.suffixIcon,
    this.prefixIcon,
    this.prefix,
    this.controller,
    this.text,
  });
}

```

```
final String? hintText;
final VoidCallback? tap;
final void Function(String)? onSubmitted;
final Widget? suffixIcon;
final Widget? prefixIcon;
final Widget? prefix;
final TextEditingController? controller;
final String? text;

@Override
Widget build(BuildContext context) {
    return TextField(
        onSubmitted: onSubmitted,
        controller: controller,
        decoration: InputDecoration(
            hintText: hintText,
            prefixIcon: prefixIcon,
            prefix: prefix,
            suffixIcon: suffixIcon,
            enabledBorder: OutlineInputBorder(
                borderSide: BorderSide(color: kPrimaryColor, width: 1.w),
                borderRadius: const BorderRadius.all(
                    Radius.circular(20),
                ).w,
            ),
            focusedBorder: OutlineInputBorder(
                borderSide: BorderSide(color: kPrimaryColor, width: 3.w),
                borderRadius: const BorderRadius.all(
                    Radius.circular(20),
                ),
            ),
        );
    }
}
```

## 5.3 Back-End

### 5.3.1 Introduction

#### Project Overview

This project focuses on the application of an e-commerce platform that aims to bridge the gap between online markets and offline stores, streamlining the buying and selling processes.

#### Technologies Used

- **ASP.NET Core:** For building the web API.
- **Entity Framework Core:** For ORM and database interactions.
- **SQL Server:** As the database system.
- **Swagger:** For API documentation and testing.

#### Architecture Overview

- **Presentation Layer:** Handles user interface and API endpoints, ensuring that user interactions are appropriately managed and responses are provided.
- **Business Logic Layer:** Contains business rules and domain logic, ensuring that the application behaves correctly according to business requirements.
- **Data Access Layer:** Interacts with the database and performs CRUD operations, ensuring data is stored and retrieved efficiently and securely.

### 5.3.2 Solution Structure

#### Layer Description

##### Presentation Layer

- Manages API endpoints and user interface interactions.
- Contains controllers, views, and front-end resources.

## **Business Logic Layer**

- Contains services and business rules.
- Implements domain logic and validation rules.

## **Data Access Layer**

- Manages database operations and interactions.
- Contains repository interfaces and implementations.

### **5.3.3 Configuration**

#### **Environment Setup**

#### **Prerequisites**

- .NET SDK
- SQL Server
- Visual Studio.

#### **Configuration Files**

##### **1. Add Connection String:**

- Open `appsettings.json` and add database connection string:

```
{  
  "ConnectionStrings": {  
    "DefaultConnection": "Data Source = OSAMA-LAPTOP\\SQLEXPRESS; Initial Catalog = OnBudget; Integrated Security = true ; Encrypt = false"  
  },  
  
  "Logging": {  
    "LogLevel": {  
      "Default": "Information",  
      "Microsoft.AspNetCore": "Warning"  
    }  
  },  
  "AllowedHosts": "*"  
}  
//Data Source=SQL8010.site4now.net;Initial Catalog=db_aa8533_onbudget;User  
Id=db_aa8533_onbudget_admin;Password=YOUR_DB_PASSWORD  
//Data Source = SQL8010.site4now.net; Initial Catalog=db_aa8533_onbudget;User  
Id=db_aa8533_onbudget_admin;Password=onbudget-1"
```

## 2. Install Entity Framework Core:

- Open a terminal in the project directory and install EF Core packages:

```
dotnet add package Microsoft.EntityFrameworkCore  
dotnet add package Microsoft.EntityFrameworkCore.SqlServer  
dotnet add package Microsoft.EntityFrameworkCore.Tools
```

## 3. Create a DbContext:

- Create a new class `ApplicationContext.cs` in the `Data` folder:

```
using Microsoft.EntityFrameworkCore;  
using OnBudget.DA.Model.Entities;  
  
namespace OnBudget.DA.AppContext  
{  
    public class ApplicationContext : DbContext  
    {  
        public ApplicationContext(DbContextOptions<ApplicationContext> options)  
            : base(options) {}  
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  
        {  
            if (!optionsBuilder.IsConfigured)  
            {  
                optionsBuilder.UseSqlServer("Data Source = OSAMA-LAPTOP\\SQLEXPRESS;  
Initial Catalog = OnBudget; Integrated Security = true ; Encrypt = false");  
            }  
            // "Data Source = SQL8010.site4now.net;Initial  
Catalog=db_aa8533_onbudget;User Id=db_aa8533_onbudget_admin;Password=onbudget-1"  
        }  
        protected override void OnModelCreating(ModelBuilder modelBuilder)  
        {  
            modelBuilder.Entity<Order>()  
                .HasMany(o => o.Products)  
                .WithMany(p => p.Orders)  
                .UsingEntity(j => j.ToTable("OrderProduct"));  
  
            base.OnModelCreating(modelBuilder);  
        }  
        public DbSet<Category> Categories { get; set; }  
        public DbSet<Customer> Customers { get; set; }  
        public DbSet<Order> Orders { get; set; }  
        public DbSet<Product> Products { get; set; }  
        public DbSet<Shipper> Shippers { get; set; }  
        public DbSet<Supplier> Suppliers { get; set; }  
        public DbSet<Picture> Pictures { get; set; }  
    }  
}
```

## 4. Configure DbContext in Startup:

- Open `Startup.cs` and configure the `ApplicationDbContext` in the `ConfigureServices` method:

```
using Microsoft.EntityFrameworkCore;
using OnBudget.BL.Services.CategoryService;
using OnBudget.BL.Services.CustomerService;
using OnBudget.BL.Services.LoginService;
using OnBudget.BL.Services.OrderService;
using OnBudget.BL.Services.PictureService;
using OnBudget.BL.Services.ProductService;
using OnBudget.BL.Services.RegistrationService;
using OnBudget.BL.Services.ShipperService;
using OnBudget.BL.Services.SupplierService;
using OnBudget.DA.AppContext;
using OnBudget.DA.Repository.CategoryRepo;
using OnBudget.DA.Repository.CustomerRepo;
using OnBudget.DA.Repository.PictureRepo;
using OnBudget.DA.Repository.ProductRepo;
using OnBudget.DA.Repository.ShipperRepo;
using OnBudget.DA.Repository.SupplierRepo;

namespace OnBudget
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            // Add services to the container.

            builder.Services.AddControllers();
            // Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
            builder.Services.AddEndpointsApiExplorer();
            builder.Services.AddSwaggerGen();
            builder.Services.AddDbContext<ApplicationDbContext>(option =>
option.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

            var app = builder.Build();

            // Configure the HTTP request pipeline.
            if (app.Environment.IsDevelopment())
            {
                app.UseSwagger();
                app.UseSwaggerUI();
            }

            app.UseHttpsRedirection();

            app.UseAuthorization();
        }
    }
}
```

```

        app.MapControllers();

    }
}

```

## Entities

Entities represent the data models for your application. They map to database tables and are managed by Entity Framework Core (EF Core).

### Category Entity

```

using System.ComponentModel.DataAnnotations;

namespace OnBudget.DA.Model.Entities
{
    public class Category
    {
        [Key]
        public string Name { get; set; }
        public string Description { get; set; }
        public ICollection<Product> Products { get; set; } = new HashSet<Product>();

    }
}

```

### Customer Entity

```

namespace OnBudget.DA.Model.Entities
{
    public class Customer
    {
        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Handle { get; set; }
        public string PhoneNumber { get; set; }
        public string Gender { get; set; }
        public string Address { get; set; }
        public string Password { get; set; }
        public ICollection<Order> Orders { get; set; } = new HashSet<Order>();
        public ICollection<Shipper> Shippers { get; set; } = new HashSet<Shipper>();
    }
}

```

```
    }  
}
```

## Order Entity

```
namespace OnBudget.DA.Model.Entities  
{  
    public class Order  
    {  
        public int Id { get; set; }  
        public DateTime OrderDate { get; set; }  
        public DateTime RequiredDate { get; set; }  
        public double TotalPrice { get; set; }  
        public int Quantity { get; set; }  
        public int CustomerId { get; set; }  
        public ICollection<Product> Products { get; set; } = new List<Product>();  
    }  
}
```

## Picture Entity

```
namespace OnBudget.DA.Model.Entities  
{  
    public class Picture  
    {  
        public int Id { get; set; }  
        public string Front { get; set; }  
        public string Back { get; set; }  
        public int ProductId { get; set; }  
    }  
}
```

## Product Entity

```
namespace OnBudget.DA.Model.Entities  
{  
    public class Product  
    {  
        public int Id { get; set; }  
        public string ProductName { get; set; }  
        public string ProductDescription { get; set; }  
        public double UnitPrice { get; set; }  
        public string Color { get; set; }  
        public string CategoryName { get; set; }  
        public string SupplierHandle { get; set; }  
    }  
}
```

```

        public ICollection<Picture> Pictures { get; set; } = new HashSet<Picture>();
        public ICollection<Order> Orders { get; set; } = new List<Order>();
    }
}

```

## Shipper Entity

```

namespace OnBudget.DA.Model.Entities
{
    public class Shipper
    {
        public int Id { get; set; }
        public string CompanyName { get; set; }
        public string Phone { get; set; }
        public int CustomerId { get; set; }
        public ICollection<Supplier> Suppliers { get; set; }
    }
}

```

## Supplier Entity

```

using System.ComponentModel.DataAnnotations;

namespace OnBudget.DA.Model.Entities
{
    public class Supplier
    {
        [Key]
        public string Handle { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string PhoneNumber { get; set; }
        public string CompanyName { get; set; }
        public string Password { get; set; }
        public ICollection<Product> Products { get; set; } = new HashSet<Product>();
    }
}

```

## Dependency Injection

### 1. Create Repository Interfaces and Implementations:

- Create interfaces and classes for repositories (e.g., IProductRepository, ProductRepository).

### 2. Create Service Interfaces and Implementations:

- Create interfaces and classes for services (e.g., `IProductService`, `ProductService`).

### 3. Register Repositories and Services in Startup:

- Open `Startup.cs` and register the repositories and services in the `ConfigureServices` method:

```
builder.Services.AddScoped<ICustomerRepository, CustomerRepository>();
builder.Services.AddScoped<ICustomerService, CustomerService>();

builder.Services.AddScoped<ISupplierService, SupplierService>();
builder.Services.AddScoped<ISupplierRepository, SupplierRepository>();

builder.Services.AddScoped<IProductRepository, ProductRepository>();
builder.Services.AddScoped<IProductService, ProductService>();

builder.Services.AddScoped<ICategoryService, CategoryService>();
builder.Services.AddScoped<ICategoryRepository, CategoryRepository>();

builder.Services.AddScoped<IOrderService, OrderService>();
builder.Services.AddScoped<IOrderRepository, OrderRepository>();

builder.Services.AddScoped<IShipperRepository, ShipperRepository>();
builder.Services.AddScoped<IShipperService, ShipperService>();

builder.Services.AddScoped<IRegistrationService, RegistrationService>();

builder.Services.AddScoped<ILoginService, LoginService>();

builder.Services.AddScoped<IPictureService, PictureService>();
builder.Services.AddScoped<IPictureRepository, PictureRepository>();
```

## Migrations

### 1. Create Migration:

- Open a terminal in the project directory and run the following commands:

```
dotnet ef migrations add InitialCreate
dotnet ef database update
```

### 2. Verify Database:

- Use your DBMS tool (e.g., SSMS for SQL Server) to verify that the database and tables have been created.

## 5.3.4 API Documentation

### Overview

APIs (Application Programming Interfaces) are sets of rules and protocols that allow different software applications to communicate and interact with each other. They define the methods and data formats that applications can use to request and exchange information. APIs enable developers to access specific features or data from external services, libraries, or systems without knowing the internal workings of those systems.

### Purpose

1. **Integration:** APIs facilitate integration between different software systems, allowing them to work together seamlessly. For example, an e-commerce website may integrate with a payment gateway API to process transactions.
2. **Data Access:** APIs provide access to data stored in remote systems or databases. This data can be retrieved, updated, or manipulated using API calls. For instance, social media platforms offer APIs to access user profiles, posts, and interactions.
3. **Functionality Extension:** APIs extend the functionality of software applications by allowing developers to add new features or services. For example, developers can integrate mapping APIs to add location-based services to their applications.
4. **Automation:** APIs enable automation by allowing programs to interact with each other programmatically. This automation can streamline processes and improve efficiency. For instance, APIs are used in DevOps for automating deployment processes.
5. **Standardization:** APIs provide standardized interfaces for accessing services or data, making it easier for developers to understand and use them across different platforms and programming languages.

## How to Use APIs

- 1. API Documentation:** Read the documentation provided by the API provider to understand how to use the API, including endpoints, request methods, parameters, and response formats.
- 2. Authentication:** Some APIs require authentication to access protected resources. Follow the authentication process specified in the documentation to obtain the necessary credentials (API keys, OAuth tokens, etc.).
- 3. API Requests:** Use HTTP methods (GET, POST, PUT, DELETE) to interact with the API endpoints. Construct API requests with appropriate parameters and headers as specified in the documentation.
- 4. Handling Responses:** Parse and handle the responses returned by the API. Responses may include success or error codes, along with the requested data formatted in JSON, XML, or other formats.
- 5. Error Handling:** Handle errors returned by the API gracefully. APIs may return error responses for various reasons such as invalid requests, authentication failures, or server errors.
- 6. Rate Limiting:** Some APIs impose rate limits to prevent abuse. Ensure that your application adheres to any rate limits specified by the API provider to avoid being blocked.
- 7. Testing:** Test your API requests using tools like Postman, cURL, or built-in browser tools. Verify that your requests are formatted correctly and that you receive the expected responses.

## Endpoints

### Category

Category	
GET	/api/Category/{CategoryName}
PUT	/api/Category/{CategoryName}
DELETE	/api/Category/{CategoryName}
POST	/api/Category
GET	/api/Category

## Customer

Customer	
GET	/api/Customer/{id}
PUT	/api/Customer/{id}
DELETE	/api/Customer/{id}
GET	/api/Customer
POST	/api/Customer

## Login

Login	
POST	/api/Login

## Order

Order	
GET	/api/Order/{id}
PUT	/api/Order/{id}
DELETE	/api/Order/{id}
GET	/api/Order
POST	/api/Order

## Pictures

Pictures	
GET	/api/Pictures
POST	/api/Pictures
PUT	/api/Pictures/{id}
DELETE	/api/Pictures/{id}

## Product

Product	
GET	/api/Product/{productName}
POST	/api/Product
GET	/api/Product
PUT	/api/Product/{id}
DELETE	/api/Product/{id}

## Registration

Registration	
POST	/api/Registration/customer
POST	/api/Registration/supplier

## Shipper

Shipper	
GET	/api/Shipper/{id}
PUT	/api/Shipper/{id}
DELETE	/api/Shipper/{id}
GET	/api/Shipper
POST	/api/Shipper

## Supplier

Supplier	
GET	/api/Supplier/{Handle}
PUT	/api/Supplier/{Handle}
DELETE	/api/Supplier/{Handle}
GET	/api/Supplier
POST	/api/Supplier

## 5.3.5 Repository Pattern

### Overview

The Repository Pattern is a design pattern that mediates data from and to the domain and data access layers. Repositories are used to encapsulate the logic required to access data sources and to centralize data access logic. They serve as an abstraction over data access, making it easier to manage data retrieval and persistence logic.

### Advantages

- Separation of Concerns:** By isolating data access logic in repositories, you can keep your business logic and data access code separate. This separation enhances maintainability and readability.
- Testability:** Repositories can be mocked or stubbed during unit tests, allowing for testing of business logic without relying on a real database.
- DRY Principle:** Data access logic is centralized in repositories, reducing code duplication. Any changes to data access can be made in one place rather than scattered across the codebase.
- Encapsulation:** Repositories encapsulate the logic needed to access data sources. This makes it easier to swap out data storage mechanisms without affecting the business logic.
- Reusability:** Repositories can be reused across different services or components, promoting code reuse.

### Interfaces

#### IProductRepository

- Method definitions for product-related data operations.

```
using OnBudget.DA.Model.Entities;  
  
namespace OnBudget.DA.Repository.ProductRepo  
{  
    public interface IProductRepository
```

```
    {
        Task<Product> GetByIdAsync(int id);
        Task AddAsync(Product product);
        Task UpdateAsync(Product product);
        Task RemoveAsync(int id);
        Task<List<Product>> GetAllProductsWithPicturesAsync();
        Task<IEnumerable<Product>> GetByNameAsync(string productName);
    }
}
```

## IOrderRepository

- Method definitions for order-related data operations.

```
using OnBudget.DA.Model.Entities;

public interface IOrderRepository
{
    Task<Order> GetByIdAsync(int id);
    Task<List<Order>> GetAllAsync();
    Task AddAsync(Order order);
    Task UpdateAsync(Order order);
    Task RemoveAsync(int id);
}
```

## ICustomerRepository

- Method definitions for customer-related data operations.

```
using OnBudget.DA.Model.Entities;

namespace OnBudget.DA.Repository.CustomerRepo
{
    public interface ICustomerRepository
    {
        Task<Customer> GetByIdAsync(int id);
        Task<IEnumerable<Customer>> GetAllAsync();
        Task AddAsync(Customer customer);
        Task UpdateAsync(Customer customer);
        Task RemoveAsync(int id);
        Task<Customer> GetByUsernameAsync(string username);
    }
}
```

## **ICategoryRepository**

- Method definitions for Category-related data operations.

```
using OnBudget.DA.Model.Entities;

namespace OnBudget.DA.Repository.CategoryRepo
{
    public interface ICategoryRepository
    {
        Task<Category> GetByNameAsync(string cname);
        Task AddAsync(Category category);
        Task UpdateAsync(Category category);
        Task RemoveAsync(string cname);
        Task<List<Category>> GetAllCategoriesWithProductsAsync();
    }
}
```

## **IPictureRepository**

- Method definitions for Picture-related data operations.

```
using OnBudget.DA.Model.Entities;

namespace OnBudget.DA.Repository.PictureRepo
{
    public interface IPictureRepository
    {
        Task<Picture> GetByIdAsync(int id);
        Task<IEnumerable<Picture>> GetAllAsync();
        Task AddAsync(Picture picture);
        Task UpdateAsync(Picture picture);
        Task RemoveAsync(int id);
    }
}
```

## **IShipperRepository**

- Method definitions for Shipper-related data operations.

```
using OnBudget.DA.Model.Entities;

namespace OnBudget.DA.Repository.ShipperRepo
{
    public interface IShipperRepository
```

```

    {
        Task<Shipper> GetByIdAsync(int id);
        Task<IEnumerable<Shipper>> GetAllAsync();
        Task AddAsync(Shipper shipper);
        Task UpdateAsync(Shipper shipper);
        Task RemoveAsync(int id);
    }
}

```

## ISupplierRepository

- Method definitions for Supplier-related data operations.

```

using OnBudget.DA.Model.Entities;

namespace OnBudget.DA.Repository.SupplierRepo
{
    public interface ISupplierRepository
    {
        Task<IEnumerable<Supplier>> GetAllAsync();
        Task<string> AddAsync(Supplier supplier);
        Task UpdateAsync(Supplier supplier);
        Task RemoveAsync(string Handle);
        Task<Supplier> GetByUsernameAsync(string username);
    }
}

```

## Implementations

### ProductRepository

- Implementation of IProductRepository.

```

using Microsoft.EntityFrameworkCore;
using OnBudget.DA.AppContext;
using OnBudget.DA.Model.Entities;

namespace OnBudget.DA.Repository.ProductRepo
{
    public class ProductRepository : IProductRepository
    {
        private readonly ApplicationDbContext _context;

        public ProductRepository(ApplicationDbContext context)
        {
            _context = context;
        }
    }
}

```

```

public async Task<Product> GetByIdAsync(int id)
{
    return await _context.Products.FindAsync(id);
}

public async Task AddAsync(Product product)
{
    _context.Products.Add(product);
    await _context.SaveChangesAsync();
}

public async Task UpdateAsync(Product product)
{
    _context.Entry(product).State = EntityState.Modified;
    await _context.SaveChangesAsync();
}

public async Task RemoveAsync(int id)
{
    var product = await _context.Products.FindAsync(id);
    if (product != null)
    {
        _context.Products.Remove(product);
        await _context.SaveChangesAsync();
    }
}
public async Task<List<Product>> GetAllProductsWithPicturesAsync()
{
    return await _context.Products
        .Include(c => c.Pictures)
        .Select(c => new Product
        {
            Id = c.Id,
            ProductName = c.ProductName,
            ProductDescription = c.ProductDescription,
            UnitPrice = c.UnitPrice,
            Color = c.Color,
            CategoryName = c.CategoryName,
            SupplierHandle = c.SupplierHandle,
            Pictures = c.Pictures.ToList()
        })
        .ToListAsync();
}

public async Task<IEnumerable<Product>> GetByNameAsync(string productName)
{
    return await _context.Products.Include(p => p.Pictures).Where(p =>
p.ProductName == productName).ToListAsync();
}
}

```

## OrderRepository

- Implementation of `IOrderRepository`.

```
using Microsoft.EntityFrameworkCore;
using OnBudget.DA.AppContext;
using OnBudget.DA.Model.Entities;

public class OrderRepository : IOrderRepository
{
    private readonly ApplicationDbContext _context;

    public OrderRepository(ApplicationDbContext context)
    {
        _context = context;
    }

    public async Task<Order> GetByIdAsync(int id)
    {
        return await _context.Orders.FindAsync(id);
    }

    public async Task<List<Order>> GetAllAsync()
    {
        return await _context.Orders
            .Include(p => p.Products).ThenInclude(c => c.Pictures)
            .Select(p => new Order
            {
                Id = p.Id,
                RequiredDate = DateTime.Now,
                OrderDate = DateTime.Now,
                Quantity = p.Quantity,
                TotalPrice = p.TotalPrice,
                CustomerId = p.CustomerId,
                Products = p.Products.ToList(),
            })
            .ToListAsync();
    }

    public async Task AddAsync(Order order)
    {
        _context.Orders.Add(order);
        await _context.SaveChangesAsync();
    }

    public async Task UpdateAsync(Order order)
    {
        _context.Entry(order).State = EntityState.Modified;
        await _context.SaveChangesAsync();
    }

    public async Task RemoveAsync(int id)
    {
```

```

        var order = await _context.Orders.FindAsync(id);
        if (order != null)
        {
            _context.Orders.Remove(order);
            await _context.SaveChangesAsync();
        }
    }
}

```

## CustomerRepository

- Implementation of `ICustomerRepository`.

```

using Microsoft.EntityFrameworkCore;
using OnBudget.DA.AppContext;
using OnBudget.DA.Model.Entities;

namespace OnBudget.DA.Repository.CustomerRepo
{
    public class CustomerRepository : ICustomerRepository
    {
        private readonly ApplicationDbContext _context;

        public CustomerRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task<Customer> GetByIdAsync(int id)
        {
            return await _context.Customers.FindAsync(id);
        }

        public async Task<IEnumerable<Customer>> GetAllAsync()
        {
            return await _context.Customers.ToListAsync();
        }

        public async Task AddAsync(Customer customer)
        {
            _context.Customers.Add(customer);
            await _context.SaveChangesAsync();
        }

        public async Task UpdateAsync(Customer customer)
        {
            _context.Entry(customer).State = EntityState.Modified;
            await _context.SaveChangesAsync();
        }

        public async Task RemoveAsync(int id)
        {
            var customer = await _context.Customers.FindAsync(id);
            if (customer != null)
            {
                _context.Customers.Remove(customer);
                await _context.SaveChangesAsync();
            }
        }
    }
}

```

```

    {
        var customer = await _context.Customers.FindAsync(id);
        if (customer != null)
        {
            _context.Customers.Remove(customer);
            await _context.SaveChangesAsync();
        }
    }

    public Task<Customer> GetByUsernameAsync(string username)
    {
        return _context.Customers.FirstOrDefaultAsync(customer => customer.Handle
== username);
    }
}
}

```

## CategoryRepository

- Implementation of `ICategoryRepository`.

```

using Microsoft.EntityFrameworkCore;
using OnBudget.DA.AppContext;
using OnBudget.DA.Model.Entities;

namespace OnBudget.DA.Repository.CategoryRepo
{
    public class CategoryRepository : ICategoryRepository
    {
        private readonly ApplicationDbContext _context;

        public CategoryRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task<Category> GetByNameAsync(string cname)
        {
            return await _context.Categories.Include(c => c.Products).ThenInclude(p
=> p.Pictures).FirstOrDefaultAsync(Category => Category.Name == cname);
        }

        public async Task AddAsync(Category category)
        {
            _context.Categories.Add(category);
            await _context.SaveChangesAsync();
        }

        public async Task UpdateAsync(Category category)
    }
}

```

```

    {
        _context.Entry(category).State = EntityState.Modified;
        await _context.SaveChangesAsync();
    }

    public async Task RemoveAsync(string cname)
    {
        var category = await _context.Categories.FindAsync(cname);
        if (category != null)
        {
            _context.Categories.Remove(category);
            await _context.SaveChangesAsync();
        }
    }

    public async Task<List<Category>> GetAllCategoriesWithProductsAsync()
    {
        return await _context.Categories
            .Include(c => c.Products).ThenInclude(p => p.Pictures)
            .Select(c => new Category
            {
                Name = c.Name,
                Description = c.Description,
                Products = c.Products.ToList()
            })
            .ToListAsync();
    }
}

```

## PictureRepository

- Implementation of `IPictureRepository`.

```

using Microsoft.EntityFrameworkCore;
using OnBudget.DA.AppContext;
using OnBudget.DA.Model.Entities;

namespace OnBudget.DA.Repository.PictureRepo
{
    public class PictureRepository : IPictureRepository
    {
        private readonly ApplicationDbContext _context;

        public PictureRepository(ApplicationDbContext context)
        {
            _context = context;
        }
        public async Task AddAsync(Picture picture)
        {
            _context.Pictures.Add(picture);
        }
    }
}

```

```

        await _context.SaveChangesAsync();
    }

    public async Task<IEnumerable<Picture>> GetAllAsync()
    {
        return await _context.Pictures.ToListAsync();
    }

    public async Task<Picture> GetByIdAsync(int id)
    {
        return await _context.Pictures.FindAsync(id);
    }

    public async Task RemoveAsync(int id)
    {
        var picture = await _context.Pictures.FindAsync(id);
        if (picture != null)
        {
            _context.Pictures.Remove(picture);
            await _context.SaveChangesAsync();
        }
    }

    public async Task UpdateAsync(Picture picture)
    {
        _context.Entry(picture).State = EntityState.Modified;
        await _context.SaveChangesAsync();
    }
}

```

## ShipperRepository

- Implementation of `IShipperRepository`.

```

using Microsoft.EntityFrameworkCore;
using OnBudget.DA.AppContext;
using OnBudget.DA.Model.Entities;

namespace OnBudget.DA.Repository.ShipperRepo
{
    public class ShipperRepository : IShipperRepository
    {
        private readonly ApplicationDbContext _context;

        public ShipperRepository(ApplicationDbContext context)
        {
            _context = context;
        }
    }
}

```

```

public async Task<Shipper> GetByIdAsync(int id)
{
    return await _context.Shippers.FindAsync(id);
}

public async Task<IEnumerable<Shipper>> GetAllAsync()
{
    return await _context.Shippers.Include(s => s.Suppliers)
        .Select(s => new Shipper
        {
            Id = s.Id,
            CompanyName = s.CompanyName,
            Phone = s.Phone,
            CustomerId = s.CustomerId,
            Suppliers = s.Suppliers.ToList(),
        })
        .ToListAsync();
}

public async Task AddAsync(Shipper shipper)
{
    _context.Shippers.Add(shipper);
    await _context.SaveChangesAsync();
}

public async Task UpdateAsync(Shipper shipper)
{
    _context.Entry(shipper).State = EntityState.Modified;
    await _context.SaveChangesAsync();
}

public async Task RemoveAsync(int id)
{
    var shipper = await _context.Shippers.FindAsync(id);
    if (shipper != null)
    {
        _context.Shippers.Remove(shipper);
        await _context.SaveChangesAsync();
    }
}
}

```

## SupplierRepository

- Implementation of `ISupplierRepository`.

```

using Microsoft.EntityFrameworkCore;
using OnBudget.DA.AppContext;
using OnBudget.DA.Model.Entities;

```

```

namespace OnBudget.DA.Repository.SupplierRepo
{
    public class SupplierRepository : ISupplierRepository
    {
        private readonly ApplicationDbContext _context;

        public SupplierRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task<Supplier> GetByIdAsync(string Handle)
        {
            return await _context.Suppliers.FindAsync(Handle);
        }

        public async Task<IEnumerable<Supplier>> GetAllAsync()
        {
            return await _context.Suppliers.ToListAsync();
        }

        public async Task<string> AddAsync(Supplier supplier)
        {
            _context.Suppliers.Add(supplier);
            await _context.SaveChangesAsync();
            return supplier.Handle;
        }

        public async Task UpdateAsync(Supplier supplier)
        {
            _context.Entry(supplier).State = EntityState.Modified;
            await _context.SaveChangesAsync();
        }

        public async Task RemoveAsync(string Handle)
        {
            var supplier = await _context.Suppliers.FindAsync(Handle);
            if (supplier != null)
            {
                _context.Suppliers.Remove(supplier);
                await _context.SaveChangesAsync();
            }
        }

        public Task<Supplier> GetByUsernameAsync(string username)
        {
            return _context.Suppliers.FirstOrDefaultAsync(Supplier => Supplier.Handle
== username);
        }
    }
}

```

## 5.3.6 Service Layer

### Overview

The service layer is a crucial part of an application's architecture that acts as an intermediary between the controllers (which handle HTTP requests) and the repositories (which manage data access). Its primary purpose is to encapsulate business logic, ensuring that the application's operations are coherent and maintainable. This layer provides a set of methods that perform operations using data retrieved or modified through repositories, ensuring that the business rules are applied consistently across the application.

### Purpose

- Encapsulation of Business Logic:** The service layer contains business logic, making the code in controllers and repositories cleaner and more focused on their specific responsibilities.
- Separation of Concerns:** By separating business logic from data access and presentation layers, the service layer enhances the maintainability and readability of the codebase.
- Reusability:** Business logic encapsulated in services can be reused across different parts of the application, reducing code duplication.
- Testability:** Services can be unit tested independently of the web layer (controllers) and the data access layer (repositories), making it easier to ensure that the business logic is correct.
- Centralized Logic:** Changes to business rules need to be made in one place only, ensuring consistency across the application.

### Interaction

#### Interaction with Repositories

- Repositories:** The service layer interacts with repositories to perform CRUD operations. It uses repository methods to fetch,

insert, update, and delete data. Repositories provide a way to abstract data access logic from the business logic.

- **Mapping:** In the absence of `AutoMapper`, the service layer manually maps domain entities to DTOs and vice versa. This manual mapping ensures that the service layer controls how data is transformed and presented to the controllers.

## Interaction with Controllers

- **Controllers:** The controllers receive HTTP requests from the client. They use the service layer to perform the required operations. The controller methods call the corresponding service methods, which in turn interact with the repositories to fetch or modify data.
- **DTOs:** Controllers use DTOs (Data Transfer Objects) to send and receive data. This helps in maintaining a clear contract between the client and the server, and ensures that only the required data is exposed to the client.

## Interfaces

### IP ProductService

- Method definitions for product-related business logic.

```
using OnBudget.BL.DTOs.PictureDtos;
using OnBudget.BL.DTOs.ProductDtos;

namespace OnBudget.BL.Services.ProductService
{
    public interface IP ProductService
    {
        Task<int> AddProductAsync(WriteProductDto productDto, WritePictureDto
pictureDto);
        Task UpdateProductAsync(int id, WriteProductDto productDto);
        Task RemoveProductAsync(int id);
        Task<List<ReadProductDto>> GetAllProductsWithPicturesAsync();
        Task<IEnumerable<ReadProductDto>> GetProductByNameAsync(string productName);
    }
}
```

## **IOrderService**

- Method definitions for Order-related business logic.

```
using OnBudget.BL.DTOs.OrderRepo;

namespace OnBudget.BL.Services.OrderService
{
    public interface IOrderService
    {
        Task<ReadOrderDto> GetOrderByIdAsync(int id);
        Task<List<ReadOrderDto>> GetAllOrdersAsync();
        Task<int> AddOrderAsync(WriteOrderDto orderDto);
        Task UpdateOrderAsync(int id, WriteOrderDto orderDto);
        Task RemoveOrderAsync(int id);
    }
}
```

## **ICustomerService**

- Method definitions for Customer-related business logic.

```
using OnBudget.BL.DTOs.CustomerDtos;

namespace OnBudget.BL.Services.CustomerService
{
    public interface ICustomerService
    {
        Task<ReadCustomerDto> GetCustomerByIdAsync(int id);
        Task<IEnumerable<ReadCustomerDto>> GetAllCustomersAsync();
        Task<int> AddCustomerAsync(WriteCustomerDto customerDto);
        Task UpdateCustomerAsync(int id, WriteCustomerDto customerDto);
        Task RemoveCustomerAsync(int id);
    }
}
```

## **ICategoryService**

- Method definitions for Category-related business logic.

```
using OnBudget.BL.DTOs.CategoryDtos;

namespace OnBudget.BL.Services.CategoryService
```

```
{  
    public interface ICategoryService  
    {  
        Task<ReadCategoryDto> GetCategoryByNameAsync(string cname);  
        Task<string> AddCategoryAsync(WriteCategoryDto writeCategoryDto);  
        Task UpdateCategoryAsync(string cname, WriteCategoryDto writeCategoryDto);  
        Task RemoveCategoryAsync(string cname);  
        Task<List<ReadCategoryDto>> GetAllCategoriesWithProductsAsync();  
    }  
}
```

## IPictureService

- Method definitions for Picture-related business logic.

```
using OnBudget.BL.DTOs.PictureDtos;  
  
namespace OnBudget.BL.Services.PictureService  
{  
    public interface IPictureService  
    {  
        Task<IEnumerable<ReadPictureDto>> GetAllPicturesAsync();  
        Task<int> AddPictureAsync(WritePictureDto writePictureDto);  
        Task UpdatePictureAsync(int id, WritePictureDto writePictureDto);  
        Task RemovePictureAsync(int id);  
    }  
}
```

## IShipperService

- Method definitions for Shipper-related business logic.

```
using OnBudget.BL.DTOs.ShipperDtos;  
  
namespace OnBudget.BL.Services.ShipperService  
{  
    public interface IShipperService  
    {  
        Task<ReadShipperDto> GetShipperByIdAsync(int id);  
        Task<IEnumerable<ReadShipperDto>> GetAllShippersAsync();  
        Task<ReadShipperDto> AddShipperAsync(WriteShipperDto shipperDto);  
        Task UpdateShipperAsync(int id, WriteShipperDto shipperDto);  
        Task RemoveShipperAsync(int id);  
    }  
}
```

## **ISupplierService**

- Method definitions for Supplier-related business logic.

```
using OnBudget.BL.DTOs.SupplierDtos;

namespace OnBudget.BL.Services.SupplierService
{
    public interface ISupplierService
    {
        Task<IEnumerable<ReadSupplierDto>> GetAllSuppliersAsync();
        Task<string> AddSupplierAsync(WriteSupplierDto supplierDto);
        Task UpdateSupplierAsync(string Handle, WriteSupplierDto supplierDto);
        Task RemoveSupplierAsync(string Handle);
        Task<ReadSupplierDto> GetByUsernameAsync(string handle);
    }
}
```

## **IRегистrationService**

- Method definitions for Registration-related business logic.

```
namespace OnBudget.BL.Services.RegistrationService
{
    public interface IRegistrationService
    {
        Task<int> RegisterCustomerAsync(WriteCustomerDto customerDto);
        Task<string> RegisterSupplierAsync(WriteSupplierDto supplierDto);
    }
}
```

## **ILoginService**

- Method definitions for Login-related business logic.

```
namespace OnBudget.BL.Services.LoginService
{
    public interface ILoginService
    {
        Task<string> LoginAsync(LoginDto loginDto, string userType);
    }
}
```

## Implementations

### ProductService

- Implementation of `IProductService`.

```
using OnBudget.BL.DTOs.PictureDtos;
using OnBudget.BL.DTOs.ProductDtos;
using OnBudget.DA.Model.Entities;
using OnBudget.DA.Repository.PictureRepo;
using OnBudget.DA.Repository.ProductRepo;

namespace OnBudget.BL.Services.ProductService
{
    public class ProductService : IProductService
    {
        private readonly IProductRepository _productRepository;
        private readonly IPictureRepository _pictureRepository;

        public ProductService(IProductRepository productRepository,
IPictureRepository pictureRepository)
        {
            _productRepository = productRepository;
            _pictureRepository = pictureRepository;
        }

        public async Task<IEnumerable<ReadProductDto>> GetProductByNameAsync(string
productName)
        {
            var products = await _productRepository.GetByNameAsync(productName);
            return products.Select(MapToDto);
        }

        public async Task<int> AddProductAsync(WriteProductDto productDto,
WritePictureDto pictureDto)
        {
            var product = new Product
            {
                ProductName = productDto.ProductName,
                ProductDescription = productDto.ProductDescription,
                UnitPrice = productDto.UnitPrice,
                Color = productDto.Color,
                SupplierHandle = productDto.SupplierHandle,
                CategoryName = productDto.CategoryName,
            };
            await _productRepository.AddAsync(product);
            var picture = new Picture
            {
                Front = pictureDto.Front,
                Back = pictureDto.Back,
                ProductId = product.Id,
            };
        }
    }
}
```

```

        await _pictureRepository.AddAsync(picture);
        return product.Id;
    }

    public async Task UpdateProductAsync(int id, WriteProductDto productDto)
    {
        var product = await _productRepository.GetByIdAsync(id);
        if (product != null)
        {
            product.ProductName = productDto.ProductName;
            product.ProductDescription = productDto.ProductDescription;
            product.UnitPrice = productDto.UnitPrice;
            product.Color = productDto.Color;
            product.SupplierHandle = productDto.SupplierHandle;
            product.CategoryName = productDto.CategoryName;
            await _productRepository.UpdateAsync(product);
        }
    }

    public async Task RemoveProductAsync(int id)
    {
        await _productRepository.RemoveAsync(id);
    }

    private static ReadProductDto MapToDto(Product product)
    {
        return new ReadProductDto
        {
            Id = product.Id,
            ProductName = product.ProductName,
            ProductDescription = product.ProductDescription,
            UnitPrice = product.UnitPrice,
            Color = product.Color,
            CategoryName = product.CategoryName,
            SupplierHandle = product.SupplierHandle,
            Pictures = product.Pictures.Select(Picture => new ReadPictureDto
            {
                Front = Picture.Front,
                Back = Picture.Back,
            }).ToList()
        };
    }

    public async Task<List<ReadProductDto>> GetAllProductsWithPicturesAsync()
    {
        var products = await
_productRepository.GetAllProductsWithPicturesAsync();
        return products.Select(products => new ReadProductDto
        {
            Id = products.Id,
            ProductName = products.ProductName,
            ProductDescription = products.ProductDescription,
            UnitPrice = products.UnitPrice,
            Color = products.Color,
        });
    }
}

```

```

        CategoryName = products.CategoryName,
        SupplierHandle = products.SupplierHandle,
        Pictures = products.Pictures.Select(Picture => new ReadPictureDto
        {
            Front = Picture.Front,
            Back = Picture.Back,
        }).ToList()
    }).ToList();
}

}

```

## OrderService

- Implementation of `IOrderService`.

```

using Microsoft.Extensions.Logging;
using OnBudget.BL.DTOs.OrderRepo;
using OnBudget.BL.DTOs.PictureDtos;
using OnBudget.BL.DTOs.ProductDtos;
using OnBudget.DA.Model.Entities;
using OnBudget.DA.Repository.ProductRepo;

namespace OnBudget.BL.Services.OrderService
{
    public class OrderService : IOrderService
    {
        private readonly IOrderRepository _orderRepository;
        private readonly IProductRepository _productRepository;
        private readonly ILogger<OrderService> _logger;

        public OrderService(IOrderRepository orderRepository, IProductRepository productRepository, ILogger<OrderService> logger)
        {
            _orderRepository = orderRepository;
            _productRepository = productRepository;
            _logger = logger;
        }

        public async Task<ReadOrderDto> GetOrderByIdAsync(int id)
        {
            var order = await _orderRepository.GetByIdAsync(id);
            return MapToDto(order);
        }

        public async Task<List<ReadOrderDto>> GetAllOrdersAsync()
        {
            var orders = await _orderRepository.GetAllAsync();

```

```

        return orders.Select(order => new ReadOrderDto
    {
        Id = order.Id,
        OrderDate = order.OrderDate,
        RequiredDate = order.RequiredDate,
        TotalPrice = order.TotalPrice,
        Quantity = order.Quantity,
        CustomerId = order.CustomerId,
        Products = order.Products.Select(product => new ReadProductDto
        {
            Id = product.Id,
            ProductName = product.ProductName,
            ProductDescription = product.ProductDescription,
            UnitPrice = product.UnitPrice,
            Color = product.Color,
            CategoryName = product.CategoryName,
            SupplierHandle = product.SupplierHandle,
            Pictures = product.Pictures.Select(Picture => new ReadPictureDto
            {
                Front = Picture.Front,
                Back = Picture.Back,
            }).ToList()
        }).ToList()
    }).ToList();
}

public async Task<int> AddOrderAsync(WriteOrderDto orderDto)
{
    var order = new Order
    {
        OrderDate = orderDto.OrderDate,
        RequiredDate = orderDto.RequiredDate,
        TotalPrice = orderDto.TotalPrice,
        Quantity = orderDto.Quantity,
        CustomerId = orderDto.CustomerId,
        Products = new List<Product>()
    };

    foreach (var productId in orderDto.ProductIds)
    {
        var product = await _productRepository.GetByIdAsync(productId);
        if (product == null)
        {
            _logger.LogError($"Product with ID {productId} not found.");
            continue;
        }

        order.Products.Add(product);
    }

    await _orderRepository.AddAsync(order);
    return order.Id;
}
public async Task UpdateOrderAsync(int id, WriteOrderDto orderDto)

```

```

    {
        var order = await _orderRepository.GetByIdAsync(id);
        if (order != null)
        {
            order.OrderDate = orderDto.OrderDate;
            order.RequiredDate = orderDto.RequiredDate;
            order.TotalPrice = orderDto.TotalPrice;
            order.Quantity = orderDto.Quantity;
            order.CustomerId = orderDto.CustomerId;
            order.Products = new List<Product>();
            foreach (var productId in orderDto.ProductIds)
            {
                var product = await _productRepository.GetByIdAsync(productId);
                if (product == null)
                {
                    _logger.LogError($"Product with ID {productId} not found.");
                    continue;
                }

                order.Products.Add(product);
            }
            await _orderRepository.UpdateAsync(order);
        }
    }

    public async Task RemoveOrderAsync(int id)
    {
        await _orderRepository.RemoveAsync(id);
    }

    private static ReadOrderDto MapToDto(Order order)
    {
        return new ReadOrderDto
        {
            Id = order.Id,
            OrderDate = order.OrderDate,
            RequiredDate = order.RequiredDate,
            TotalPrice = order.TotalPrice,
            Quantity = order.Quantity,
            CustomerId = order.CustomerId,
        };
    }
}

```

## CustomerService

- Implementation of `ICustomerService`.

```
using OnBudget.BL.DTOs.CustomerDtos;
using OnBudget.DA.Model.Entities;
using OnBudget.DA.Repository.CustomerRepo;

namespace OnBudget.BL.Services.CustomerService
{
    public class CustomerService : ICustomerService
    {
        private readonly ICustomerRepository _customerRepository;

        public CustomerService(ICustomerRepository customerRepository)
        {
            _customerRepository = customerRepository;
        }

        public async Task<ReadCustomerDto> GetCustomerByIdAsync(int id)
        {
            var customer = await _customerRepository.GetByIdAsync(id);
            return MapToDto(customer);
        }

        public async Task<IEnumerable<ReadCustomerDto>> GetAllCustomersAsync()
        {
            var customers = await _customerRepository.GetAllAsync();
            return customers.Select(MapToDto);
        }

        public async Task<int> AddCustomerAsync(WriteCustomerDto customerDto)
        {
            var customer = new Customer
            {
                FirstName = customerDto.FirstName,
                LastName = customerDto.LastName,
                Handle = customerDto.Handle,
                PhoneNumber = customerDto.PhoneNumber,
                Gender = customerDto.Gender,
                Address = customerDto.Address,
                Password = customerDto.Password,
            };
            await _customerRepository.AddAsync(customer);
            return customer.Id;
        }

        public async Task UpdateCustomerAsync(int id, WriteCustomerDto customerDto)
        {
            var customer = await _customerRepository.GetByIdAsync(id);
            if (customer != null)
            {
                customer.FirstName = customerDto.FirstName;
            }
        }
    }
}
```

```

        customer.LastName = customerDto.LastName;
        customer.Handle = customerDto.Handle;
        customer.PhoneNumber = customerDto.PhoneNumber;
        customer.Gender = customerDto.Gender;
        customer.Address = customerDto.Address;
        customer.Password = customerDto.Password;
        await _customerRepository.UpdateAsync(customer);
    }
}

public async Task RemoveCustomerAsync(int id)
{
    await _customerRepository.RemoveAsync(id);
}

private static ReadCustomerDto MapToDto(Customer customer)
{
    return new ReadCustomerDto
    {
        Id = customer.Id,
        FirstName = customer.FirstName,
        LastName = customer.LastName,
        Handle = customer.Handle,
        PhoneNumber = customer.PhoneNumber,
        Gender = customer.Gender,
        Address = customer.Address
    };
}
}

```

## CategoryService

- Implementation of `ICategoryService`.

```

using OnBudget.BL.DTOs.CategoryDtos;
using OnBudget.BL.DTOs.PictureDtos;
using OnBudget.BL.DTOs.ProductDtos;
using OnBudget.DA.Model.Entities;
using OnBudget.DA.Repository.CategoryRepo;

namespace OnBudget.BL.Services.CategoryService
{
    public class CategoryService : ICategoryService
    {
        private readonly ICategoryRepository _categoryRepository;

        public CategoryService(ICategoryRepository categoryRepository)
        {
            _categoryRepository = categoryRepository;
        }
    }
}

```

```

public async Task<ReadCategoryDto> GetCategoryByNameAsync(string name)
{
    var category = await _categoryRepository.GetByNameAsync(name);
    return MapToDto(category);
}

public async Task<string> AddCategoryAsync(WriteCategoryDto writeCategoryDto)
{
    var category = new Category
    {
        Name = writeCategoryDto.Name,
        Description = writeCategoryDto.Description
    };
    await _categoryRepository.AddAsync(category);
    return category.Name;
}

public async Task UpdateCategoryAsync(string name, WriteCategoryDto
writeCategoryDto)
{
    var category = await _categoryRepository.GetByNameAsync(name);
    if (category != null)
    {
        category.Name = writeCategoryDto.Name;
        category.Description = writeCategoryDto.Description;
        await _categoryRepository.UpdateAsync(category);
    }
}

public async Task RemoveCategoryAsync(string cname)
{
    await _categoryRepository.RemoveAsync(cname);
}

private static ReadCategoryDto MapToDto(Category category)
{
    return new ReadCategoryDto
    {
        Name = category.Name,
        Description = category.Description,
        Products= (List<ReadProductDto>)category.Products.Select(product =>
new ReadProductDto
{
    Id = product.Id,
    ProductName = product.ProductName,
    ProductDescription = product.ProductDescription,
    UnitPrice = product.UnitPrice,
    Color = product.Color,
    CategoryName = product.CategoryName,
    SupplierHandle = product.SupplierHandle,
    Pictures = product.Pictures.Select(Picture => new ReadPictureDto
{
    Front = Picture.Front,
    Back = Picture.Back,
}
)
    }
}
}

```

```
        }).ToList()
    }).ToList()

};

}

public async Task<List<ReadCategoryDto>> GetAllCategoriesWithProductsAsync()
{
    var categories = await
_categoryRepository.GetAllCategoriesWithProductsAsync();
    return categories.Select(category => new ReadCategoryDto
{
    Name = category.Name,
    Description = category.Description,
    Products = category.Products.Select(product => new ReadProductDto
{
    Id = product.Id,
    ProductName = product.ProductName,
    ProductDescription = product.ProductDescription,
    UnitPrice = product.UnitPrice,
    Color = product.Color,
    CategoryName = product.CategoryName,
    SupplierHandle = product.SupplierHandle,
    Pictures = product.Pictures.Select(Picture => new ReadPictureDto
{
        Front = Picture.Front,
        Back = Picture.Back,

        }).ToList()
    }).ToList()
}).ToList();
}
```

# PictureService

- Implementation of `IPictureService`.

```
using OnBudget.BL.DTOs.PictureDtos;
using OnBudget.DA.Model.Entities;
using OnBudget.DA.Repository.PictureRepo;

namespace OnBudget.BL.Services.PictureService
{
    public class PictureService : IPictureService
    {
        private readonly IPictureRepository _pictureRepository;

        public PictureService(IPictureRepository pictureRepository)
        {
            pictureRepository = pictureRepository;
        }
    }
}
```

```

    }
    public async Task<int> AddPictureAsync(WritePictureDto writePictureDto)
    {
        var picture = new Picture
        {
            Front = writePictureDto.Front,
            Back = writePictureDto.Back,
        };
        await _pictureRepository.AddAsync(picture);
        return picture.Id;
    }

    public async Task<IEnumerable<ReadPictureDto>> GetAllPicturesAsync()
    {
        var pictures = await _pictureRepository.GetAllAsync();
        return pictures.Select(MapToDto);
    }

    public async Task RemovePictureAsync(int id)
    {
        await _pictureRepository.RemoveAsync(id);
    }

    public async Task UpdatePictureAsync(int id, WritePictureDto writePictureDto)
    {
        var picture = await _pictureRepository.GetByIdAsync(id);
        if (picture != null)
        {
            picture.Front = writePictureDto.Front;
            picture.Back = writePictureDto.Back;
            await _pictureRepository.UpdateAsync(picture);
        }
    }
    private static ReadPictureDto MapToDto(Picture picture)
    {
        return new ReadPictureDto
        {
            Front = picture.Front,
            Back = picture.Back,
        };
    }
}

```

## ShipperService

- Implementation of `IShipperService`.

```

using OnBudget.BL.DTOs.ShipperDtos;
using OnBudget.DA.Model.Entities;
using OnBudget.DA.Repository.ShipperRepo;
using OnBudget.DA.Repository.SupplierRepo;

```

```

namespace OnBudget.BL.Services.ShipperService
{
    public class ShipperService : IShipperService
    {
        private readonly IShipperRepository _shipperRepository;
        private readonly ISupplierRepository _supplierRepository;

        public ShipperService(IShipperRepository shipperRepository,
ISupplierRepository supplierRepository)
        {
            _shipperRepository = shipperRepository;
            _supplierRepository = supplierRepository;
        }

        public async Task<ReadShipperDto> GetShipperByIdAsync(int id)
        {
            var shipper = await _shipperRepository.GetByIdAsync(id);
            return MapToDto(shipper);
        }

        public async Task<IEnumerable<ReadShipperDto>> GetAllShippersAsync()
        {
            var shippers = await _shipperRepository.GetAllAsync();
            return shippers.Select(MapToDto);
        }

        public async Task<ReadShipperDto> AddShipperAsync(WriteShipperDto shipperDto)
        {
            var shipper = new Shipper
            {
                CompanyName = shipperDto.CompanyName,
                Phone = shipperDto.Phone,
                CustomerId = shipperDto.CustomerId,
            };
            if (shipper.Suppliers == null)
            {
                shipper.Suppliers = new List<Supplier>();
            }
            foreach (var supplierHandle in shipperDto.Suppliers)
            {
                var supplier = await
_supplierRepository.GetByUsernameAsync(supplierHandle);
                if (supplier != null)
                {
                    shipper.Suppliers.Add(supplier);
                }
                else
                {
                    throw new ArgumentException($"Supplier with handle
'{supplierHandle}' not found.");
                }
            }
            await _shipperRepository.AddAsync(shipper);
        }
    }
}

```

```

        return MapToDto(shipper);
    }

    public async Task UpdateShipperAsync(int id, WriteShipperDto shipperDto)
    {
        var shipper = await _shipperRepository.GetByIdAsync(id);
        if (shipper != null)
        {
            shipper.CompanyName = shipperDto.CompanyName;
            shipper.Phone = shipperDto.Phone;
            shipper.CustomerId = shipperDto.CustomerId;

            if (shipperDto.Suppliers != null && shipperDto.Suppliers.Any())
            {
                foreach (var supplierHandle in shipperDto.Suppliers)
                {
                    var supplier = await
                    _supplierRepository.GetByUsernameAsync(supplierHandle);
                    if (supplier != null)
                    {
                        shipper.Suppliers.Add(supplier);
                    }
                    else
                    {
                        throw new ArgumentException($"Supplier with handle
'{supplierHandle}' not found.");
                    }
                }
            }
            await _shipperRepository.UpdateAsync(shipper);
        }
    }

    public async Task RemoveShipperAsync(int id)
    {
        await _shipperRepository.RemoveAsync(id);
    }

    private static ReadShipperDto MapToDto(Shipper shipper)
    {
        return new ReadShipperDto
        {
            Id = shipper.Id,
            CompanyName = shipper.CompanyName,
            Phone = shipper.Phone,
            CustomerId = shipper.CustomerId,
            Suppliers = shipper.Suppliers.Select(supplier =>
            supplier.Handle).ToList()
        };
    }
}

```

## SupplierService

- Implementation of `ISupplierService`.

```
using OnBudget.BL.DTOs.SupplierDtos;
using OnBudget.DA.Model.Entities;
using OnBudget.DA.Repository.SupplierRepo;

namespace OnBudget.BL.Services.SupplierService
{
    public class SupplierService : ISupplierService
    {
        private readonly ISupplierRepository _supplierRepository;

        public SupplierService(ISupplierRepository supplierRepository)
        {
            _supplierRepository = supplierRepository;
        }

        public async Task<ReadSupplierDto> GetByUsernameAsync(string Handle)
        {
            var supplier = await _supplierRepository.GetByUsernameAsync(Handle);
            return MapToDto(supplier);
        }

        public async Task<IEnumerable<ReadSupplierDto>> GetAllSuppliersAsync()
        {
            var suppliers = await _supplierRepository.GetAllAsync();
            return suppliers.Select(MapToDto);
        }

        public async Task<string> AddSupplierAsync(WriteSupplierDto supplierDto)
        {
            var supplier = new Supplier
            {
                FirstName = supplierDto.FirstName,
                LastName = supplierDto.LastName,
                Handle = supplierDto.Handle,
                PhoneNumber = supplierDto.PhoneNumber,
                CompanyName = supplierDto.CompanyName,
                Password = supplierDto.Password,
            };

            await _supplierRepository.AddAsync(supplier);

            return supplier.Handle;
        }

        public async Task UpdateSupplierAsync(string Handle, WriteSupplierDto
supplierDto)
        {
            var supplier = await _supplierRepository.GetByUsernameAsync(Handle);
            if (supplier != null)
```

```

        {
            supplier.FirstName = supplierDto.FirstName;
            supplier.LastName = supplierDto.LastName;
            supplier.Handle = supplierDto.Handle;
            supplier.PhoneNumber = supplierDto.PhoneNumber;
            supplier.CompanyName = supplierDto.CompanyName;
            supplier.Password = supplierDto.Password;
            await _supplierRepository.UpdateAsync(supplier);
        }
    }

    public async Task RemoveSupplierAsync(string Handle)
    {
        await _supplierRepository.RemoveAsync(Handle);
    }

    private static ReadSupplierDto MapToDto(Supplier supplier)
    {
        return new ReadSupplierDto
        {
            FirstName = supplier.FirstName,
            LastName = supplier.LastName,
            Handle = supplier.Handle,
            PhoneNumber = supplier.PhoneNumber,
            CompanyName = supplier.CompanyName
        };
    }
}

```

## RegistrationService

- Implementation of `IRegistrationService`.

```

using OnBudget.BL.DTOs.CustomerDtos;
using OnBudget.BL.DTOs.SupplierDtos;
using OnBudget.BL.Services.CustomerService;
using OnBudget.BL.Services.SupplierService;

namespace OnBudget.BL.Services.RegistrationService
{
    public class RegistrationService : IRegistrationService
    {
        private readonly ICustomerService _customerService;
        private readonly ISupplierService _supplierService;

        public RegistrationService(ICustomerService customerService, ISupplierService
supplierService)
        {
            _customerService = customerService;
            _supplierService = supplierService;
        }
    }
}

```

```

        }

        public async Task<int> RegisterCustomerAsync(WriteCustomerDto customerDto)
        {
            return await _customerService.AddCustomerAsync(customerDto);
        }

        public async Task<string> RegisterSupplierAsync(WriteSupplierDto supplierDto)
        {
            return await _supplierService.AddSupplierAsync(supplierDto);
        }
    }
}

```

## LoginService

- Implementation of `ILoginService`.

```

using OnBudget.BL.DTOs.LoginDtos;
using OnBudget.DA.Repository.CustomerRepo;
using OnBudget.DA.Repository.SupplierRepo;

namespace OnBudget.BL.Services.LoginService
{
    public class LoginService : ILoginService
    {
        private readonly ICustomerRepository _customerRepository;
        private readonly ISupplierRepository _supplierRepository;

        public LoginService(ICustomerRepository customerRepository,
ISupplierRepository supplierRepository)
        {
            _customerRepository = customerRepository;
            _supplierRepository = supplierRepository;
        }

        public async Task<string> LoginAsync(LoginDto loginDto, string userType)
        {
            if (userType.Equals("customer", StringComparison.OrdinalIgnoreCase))
            {
                return await CustomerLoginAsync(loginDto) ?
GenerateToken(loginDto.Username, userType) : null;
            }
            else if (userType.Equals("supplier", StringComparison.OrdinalIgnoreCase))
            {
                return await SupplierLoginAsync(loginDto) ?
GenerateToken(loginDto.Username, userType) : null;
            }
            else
            {
                throw new ArgumentException("Invalid user type");
            }
        }
    }
}

```

```

        }

    private async Task<bool> CustomerLoginAsync(LoginDto loginDto)
    {
        var customer = await
_customerRepository.GetByUsernameAsync(loginDto.Username);
        return customer != null && IsPasswordValid(customer.Password,
loginDto.Password);
    }

    private async Task<bool> SupplierLoginAsync(LoginDto loginDto)
    {
        var supplier = await
_supplierRepository.GetByUsernameAsync(loginDto.Username);
        return supplier != null && IsPasswordValid(supplier.Password,
loginDto.Password);
    }

    private bool IsPasswordValid(string storedPassword, string enteredPassword)
    {
        return storedPassword == enteredPassword;
    }

    private string GenerateToken(string username, string userType)
    {
        return $"token_for_{username}_as_{userType}";
    }
}

```

## 5.3.7 DTOs (Data Transfer Objects)

### Read DTOs

#### Overview

A Read DTO is a specialized Data Transfer Object used for transferring data from the server to the client, specifically for read operations. It represents a data structure that encapsulates the necessary information needed by the client to display or process the data without exposing the internal implementation details or the full structure of the domain entities.

## Purpose

1. **Abstraction:** Read DTOs abstract the underlying domain model, ensuring that clients only receive the data they need and protecting the integrity of the domain model.
2. **Security:** By using DTOs, sensitive information within domain entities can be excluded from being sent to clients, enhancing security.
3. **Performance:** DTOs can be tailored to include only the relevant data fields, reducing the amount of data transferred over the network and improving performance.
4. **Decoupling:** DTOs decouple the client interface from the internal data models, allowing changes in the domain model without affecting the client-side code.
5. **Ease of Use:** DTOs can be designed to meet the specific needs of the client, making it easier to work with data in client applications.

## Implementations

### ReadCategoryDto

```
using OnBudget.BL.DTOs.ProductDtos;

namespace OnBudget.BL.DTOs.CategoryDtos
{
    public class ReadCategoryDto
    {
        public string Name { get; set; }
        public string Description { get; set; }
        public List<ReadProductDto> Products { get; set; }
    }
}
```

### ReadCustomerDto

```
namespace OnBudget.BL.DTOs.CustomerDtos
{
    public class ReadCustomerDto
    {
        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
    }
}
```

```
        public string Handle { get; set; }
        public string PhoneNumber { get; set; }
        public string Gender { get; set; }
        public string Address { get; set; }
    }
}
```

## ReadOrderDto

```
using OnBudget.BL.DTOs.ProductDtos;

namespace OnBudget.BL.DTOs.OrderRepo
{
    public class ReadOrderDto
    {
        public int Id { get; set; }
        public DateTime OrderDate { get; set; }
        public DateTime RequiredDate { get; set; }
        public double TotalPrice { get; set; }
        public int Quantity { get; set; }
        public int CustomerId { get; set; }
        public List<ReadProductDto> Products { get; set; }
    }
}
```

## ReadPictureDto

```
namespace OnBudget.BL.DTOs.PictureDtos
{
    public class ReadPictureDto
    {
        public string Front { get; set; }
        public string Back { get; set; }
    }
}
```

## ReadProductDto

```
using OnBudget.BL.DTOs.PictureDtos;

namespace OnBudget.BL.DTOs.ProductDtos
{
    public class ReadProductDto
    {
        public int Id { get; set; }
    }
}
```

```
        public string ProductName { get; set; }
        public string ProductDescription { get; set; }
        public double UnitPrice { get; set; }
        public string Color { get; set; }
        public string CategoryName { get; set; }
        public string SupplierHandle { get; set; }
        public List<ReadPictureDto> Pictures { get; set; }
    }
}
```

## ReadShipperDto

```
namespace OnBudget.BL.DTOs.ShipperDtos
{
    public class ReadShipperDto
    {
        public int Id { get; set; }
        public string CompanyName { get; set; }
        public string Phone { get; set; }
        public int CustomerId { get; set; }
        public List<string> Suppliers { get; set; }
    }
}
```

## ReadSupplierDto

```
namespace OnBudget.BL.DTOs.SupplierDtos
{
    public class ReadSupplierDto
    {
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Handle { get; set; }
        public string PhoneNumber { get; set; }
        public string CompanyName { get; set; }
    }
}
```

# Write DTOs

## Overview

A Write DTO is a specialized Data Transfer Object used for transferring data from the client to the server, specifically for create and update operations. It represents the structure of the data that the client needs to send to the server to perform these operations, ensuring that the necessary information is provided without exposing the internal implementation details or the full structure of the domain entities.

## Purpose

- Abstraction:** Write DTOs abstract the underlying domain model, ensuring that clients send only the data needed for specific operations without accessing the full domain entities.
- Validation:** They provide a structure that can be validated to ensure that the data sent by the client meets the required criteria before any business logic or data access operations are performed.
- Security:** Write DTOs help to prevent over-posting attacks by including only the fields that are required for the operation, reducing the risk of unwanted data modifications.
- Decoupling:** DTOs decouple the client interface from the internal data models, allowing changes in the domain model without affecting the client-side code.
- Ease of Use:** DTOs can be designed to meet the specific needs of the client, making it easier to send data in a format that the server expects.

## Implementations

### WriteCategoryDto

```
namespace OnBudget.BL.DTOs.CategoryDtos
{
    public class WriteCategoryDto
    {
        public string Name { get; set; }
```

```
        public string Description { get; set; }
    }
}
```

## WriteCustomerDto

```
namespace OnBudget.BL.DTOs.CustomerDtos
{
    public class WriteCustomerDto
    {
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Handle { get; set; }
        public string PhoneNumber { get; set; }
        public string Gender { get; set; }
        public string Address { get; set; }
        public string Password { get; set; }
    }
}
```

## WriteOrderDto

```
namespace OnBudget.BL.DTOs.OrderRepo
{
    public class WriteOrderDto
    {
        public DateTime OrderDate { get; set; }
        public DateTime RequiredDate { get; set; }
        public double TotalPrice { get; set; }
        public int Quantity { get; set; }
        public int CustomerId { get; set; }
        public List<int> ProductIds { get; set; } = new List<int>();
    }
}
```

## WritePictureDto

```
namespace OnBudget.BL.DTOs.PictureDtos
{
    public class WritePictureDto
    {
        public string Front { get; set; }
        public string Back { get; set; }
    }
}
```

## **WriteProductDto**

```
namespace OnBudget.BL.DTOs.ProductDtos
{
    public class WriteProductDto
    {
        public string ProductName { get; set; }
        public string ProductDescription { get; set; }
        public double UnitPrice { get; set; }
        public string Color { get; set; }
        public string SupplierHandle { get; set; }
        public string CategoryName { get; set; }
    }
}
```

## **WriteShipperDto**

```
namespace OnBudget.BL.DTOs.ShipperDtos
{
    public class WriteShipperDto
    {
        public string CompanyName { get; set; }
        public string Phone { get; set; }
        public int CustomerId { get; set; }
        public List<string> Suppliers { get; set; }
    }
}
```

## **WriteSupplierDto**

```
namespace OnBudget.BL.DTOs.SupplierDtos
{
    public class WriteSupplierDto
    {
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Handle { get; set; }
        public string PhoneNumber { get; set; }
        public string CompanyName { get; set; }
        public string Password { get; set; }
        public int ProductId { get; set; }
    }
}
```

## **ProductPictureDto**

```
using OnBudget.BL.DTOs.PictureDtos;
using OnBudget.BL.DTOs.ProductDtos;

namespace OnBudget.BL.DTOs
{
    public class ProductPictureDto
    {
        public WriteProductDto Product { get; set; }
        public WritePictureDto Picture { get; set; }
    }
}
```

```
}
```

## LoginDto

```
namespace OnBudget.BL.DTOs.LoginDtos
{
    public class LoginDto
    {
        public string Username { get; set; }
        public string Password { get; set; }
    }
}
```

### 5.3.8 Controllers

#### Overview

Controllers in ASP.NET Core act as intermediaries between the client and the application's business logic. They handle incoming HTTP requests, execute the corresponding business logic, and return HTTP responses. In an API-driven application, controllers primarily handle API requests and return appropriate responses, usually in JSON format.

#### Purpose

- 1. Request Handling:** Controllers handle incoming HTTP requests from clients. They parse the request data, extract parameters, and determine the action to be taken based on the request method (GET, POST, PUT, DELETE, etc.) and route.
- 2. Business Logic Invocation:** Controllers invoke appropriate methods or services to execute the business logic required to fulfill the request. They interact with services, which encapsulate the application's business rules.
- 3. Data Validation:** Controllers validate incoming data from the client to ensure it meets the required criteria before processing it further. This helps maintain data integrity and prevents invalid data from entering the system.

4. **Response Generation:** After executing the necessary business logic, controllers generate and return appropriate HTTP responses to the client. Responses may include success or error messages, along with any requested data.
5. **Content Negotiation:** Controllers handle content negotiation, determining the format of the response data based on the client's preferences (JSON, XML, etc.). In API development, JSON is commonly used for data interchange.

## Handling API Requests and Returning Responses

### CategoryController

```
using Microsoft.AspNetCore.Mvc;
using OnBudget.BL.DTOs.CategoryDtos;
using OnBudget.BL.Services.CategoryService;

namespace OnBudget.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CategoryController : ControllerBase
    {
        private readonly ICategoryService _categoryService;

        public CategoryController(ICategoryService categoryService)
        {
            _categoryService = categoryService;
        }

        [HttpGet("{CategoryName}")]
        public async Task<ActionResult<ReadCategoryDto>>
GetCategoryByNameAsync(string CategoryName)
        {
            var categoryDto = await
_categoryService.GetCategoryByNameAsync(CategoryName);
            if (categoryDto == null)
            {
                return NotFound();
            }
            return Ok(categoryDto);
        }

        [HttpPost]
    }
}
```

```

        public async Task<ActionResult<string>> AddCategory(WriteCategoryDto
categoryDto)
    {
        var CategoryName = await _categoryService.AddCategoryAsync(categoryDto);
        return Ok(CategoryName);
    }

    [HttpPut("{CategoryName}")]
    public async Task<IActionResult> UpdateCategory(string CategoryName,
WriteCategoryDto categoryDto)
    {
        await _categoryService.UpdateCategoryAsync(CategoryName, categoryDto);
        return Ok();
    }

    [HttpDelete("{CategoryName}")]
    public async Task<IActionResult> RemoveCategory(string CategoryName)
    {
        await _categoryService.RemoveCategoryAsync(CategoryName);
        return Ok();
    }
    [HttpGet]
    public async Task<ActionResult<List<ReadCategoryDto>>>
GetAllCategoriesWithProducts()
    {
        var categories = await
_categoryService.GetAllCategoriesWithProductsAsync();
        return Ok(categories);
    }
}
}

```

## CustomerController

```

using Microsoft.AspNetCore.Mvc;
using OnBudget.BL.DTOs.CustomerDtos;
using OnBudget.BL.Services.CustomerService;

namespace OnBudget.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CustomerController : ControllerBase
    {
        private readonly ICustomerService _customerService;

        public CustomerController(ICustomerService customerService)
        {
            _customerService = customerService;
        }

        [HttpGet("{id}")]

```

```

public async Task<ActionResult<ReadCustomerDto>> GetCustomerById(int id)
{
    var customerDto = await _customerService.GetCustomerByIdAsync(id);
    if (customerDto == null)
    {
        return NotFound();
    }
    return Ok(customerDto);
}

[HttpGet]
public async Task<ActionResult<IEnumerable<ReadCustomerDto>>>
GetAllCustomers()
{
    var customersDto = await _customerService.GetAllCustomersAsync();
    return Ok(customersDto);
}

[HttpPost]
public async Task<ActionResult<int>> AddCustomer(WriteCustomerDto
customerDto)
{
    var customerId = await _customerService.AddCustomerAsync(customerDto);
    return CreatedAtAction(nameof(GetCustomerById), new { id = customerId },
customerId);
}

[HttpPut("{id}")]
public async Task<IActionResult> UpdateCustomer(int id, WriteCustomerDto
customerDto)
{
    await _customerService.UpdateCustomerAsync(id, customerDto);
    return NoContent();
}

[HttpDelete("{id}")]
public async Task<IActionResult> RemoveCustomer(int id)
{
    await _customerService.RemoveCustomerAsync(id);
    return NoContent();
}
}

```

## LoginController

```

using Microsoft.AspNetCore.Mvc;
using OnBudget.BL.DTOs.LoginDtos;
using OnBudget.BL.Services.LoginService;
using System.ComponentModel.DataAnnotations;

namespace OnBudget.Controllers

```

```

{
    [Route("api/[controller]")]
    [ApiController]
    public class LoginController : ControllerBase
    {
        private readonly ILoginService _loginService;

        public LoginController(ILoginService loginService)
        {
            _loginService = loginService;
        }

        [HttpPost]
        public async Task<ActionResult<string>> Login(LoginDto loginDto, [Required]
string userType)
        {
            try
            {
                var token = await _loginService.LoginAsync(loginDto, userType);
                if (token == null)
                    return Unauthorized();
                return Ok(token);
            }
            catch (ArgumentException)
            {
                return BadRequest("Invalid user type");
            }
        }
    }
}

```

## OrderController

```

using Microsoft.AspNetCore.Mvc;
using OnBudget.BL.DTOs.OrderRepo;
using OnBudget.BL.Services.OrderService;

namespace OnBudget.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class OrderController : ControllerBase
    {
        private readonly IOrderService _orderService;

        public OrderController(IOrderService orderService)
        {
            _orderService = orderService;
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<ReadOrderDto>> GetOrderById(int id)

```

```

    {
        var orderDto = await _orderService.GetOrderByIdAsync(id);
        if (orderDto == null)
        {
            return NotFound();
        }
        return Ok(orderDto);
    }

    [HttpGet]
    public async Task<ActionResult<IEnumerable<ReadOrderDto>>> GetAllOrders()
    {
        var ordersDto = await _orderService.GetAllOrdersAsync();
        return Ok(ordersDto);
    }

    [HttpPost]
    public async Task<ActionResult<int>> AddOrder(WriteOrderDto orderDto)
    {
        var orderId = await _orderService.AddOrderAsync(orderDto);
        return CreatedAtAction(nameof(GetOrderById), new { id = orderId },
orderId);
    }

    [HttpPut("{id}")]
    public async Task<IActionResult> UpdateOrder(int id, WriteOrderDto orderDto)
    {
        await _orderService.UpdateOrderAsync(id, orderDto);
        return NoContent();
    }

    [HttpDelete("{id}")]
    public async Task<IActionResult> RemoveOrder(int id)
    {
        await _orderService.RemoveOrderAsync(id);
        return NoContent();
    }
}

```

## PicturesController

```

using Microsoft.AspNetCore.Mvc;
using OnBudget.BL.DTOs.PictureDtos;
using OnBudget.BL.Services.PictureService;

namespace OnBudget.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class PicturesController : ControllerBase
    {

```

```

private readonly IPictureService _pictureService;

public PicturesController(IPictureService pictureService)
{
    _pictureService = pictureService;
}

[HttpGet]
public async Task<ActionResult<IEnumerable<ReadPictureDto>>>
GetAllPicturesAsync()
{
    var pictures = await _pictureService.GetAllPicturesAsync();
    return Ok(pictures);
}

[HttpPost]
public async Task<ActionResult<int>> AddPictureAsync(WritePictureDto
writePictureDto)
{
    var pictureId = await _pictureService.AddPictureAsync(writePictureDto);
    return Ok(pictureId);
}

[HttpPut("{id}")]
public async Task<ActionResult> UpdatePictureAsync(int id, WritePictureDto
writePictureDto)
{
    await _pictureService.UpdatePictureAsync(id, writePictureDto);
    return NoContent();
}

[HttpDelete("{id}")]
public async Task<ActionResult> RemovePictureAsync(int id)
{
    await _pictureService.RemovePictureAsync(id);
    return NoContent();
}
}

```

## ProductController

```

using Microsoft.AspNetCore.Mvc;
using OnBudget.BL.DTOs;
using OnBudget.BL.DTOs.ProductDtos;
using OnBudget.BL.Services.ProductService;

namespace OnBudget.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ProductController : ControllerBase

```

```

{
    private readonly IProductService _ productService;

    public ProductController(IProductService productService)
    {
        _ productService = productService;
    }

    [HttpGet("{productName}")]
    public async Task<IActionResult> GetProductsByName(string productName)
    {
        try
        {
            var products = await
_productService.GetProductByNameAsync(productName);
            if (products == null || !products.Any())
            {
                return NotFound();
            }
            return Ok(products);
        }
        catch (Exception ex)
        {
            return StatusCode(500, $"Internal server error: {ex.Message}");
        }
    }

    [HttpPost]
    public async Task<IActionResult> AddProductAndPicture([FromBody]
ProductPictureDto dto)
    {
        if (dto == null)
        {
            return BadRequest("Invalid request body");
        }

        int productId = await _ productService.AddProductAsync(dto.Product,
dto.Picture);

        return Ok(productId);
    }

    [HttpPut("{id}")]
    public async Task<IActionResult> UpdateProduct(int id, WriteProductDto
productDto)
    {
        await _ productService.UpdateProductAsync(id, productDto);
        return NoContent();
    }

    [HttpDelete("{id}")]
    public async Task<IActionResult> RemoveProduct(int id)
    {
}

```

```

        await _productService.RemoveProductAsync(id);
        return NoContent();
    }
    [HttpGet]
    public async Task<ActionResult<List<ReadProductDto>>>
GetAllProductsWithPicturesAsync()
{
    var products = await _productService.GetAllProductsWithPicturesAsync();
    return Ok(products);
}
}
}

```

## RegistrationController

```

using Microsoft.AspNetCore.Mvc;
using OnBudget.BL.DTOs.CustomerDtos;
using OnBudget.BL.DTOs.SupplierDtos;
using OnBudget.BL.Services.RegistrationService;

namespace OnBudget.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class RegistrationController : ControllerBase
    {
        private readonly IRegistrationService _registrationService;

        public RegistrationController(IRegistrationService registrationService)
        {
            _registrationService = registrationService;
        }

        [HttpPost("customer")]
        public async Task<ActionResult<int>> RegisterCustomer(WriteCustomerDto
customerDto)
        {
            var customerId = await
_registrationService.RegisterCustomerAsync(customerDto);
            return CreatedAtAction("RegisterCustomer", new { id = customerId },
customerId);
        }

        [HttpPost("supplier")]
        public async Task<ActionResult<int>> RegisterSupplier(WriteSupplierDto
supplierDto)
        {
            var supplierId = await
_registrationService.RegisterSupplierAsync(supplierDto);
            return CreatedAtAction("RegisterSupplier", new { id = supplierId },
supplierId);
        }
    }
}

```

## ShipperController

```
using Microsoft.AspNetCore.Mvc;
using OnBudget.BL.DTOs.ShipperDtos;
using OnBudget.BL.Services.ShipperService;

namespace OnBudget.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ShipperController : ControllerBase
    {
        private readonly IShipperService _shipperService;

        public ShipperController(IShipperService shipperService)
        {
            _shipperService = shipperService;
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<ReadShipperDto>> GetShipperById(int id)
        {
            var shipperDto = await _shipperService.GetShipperByIdAsync(id);
            if (shipperDto == null)
            {
                return NotFound();
            }
            return Ok(shipperDto);
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<ReadShipperDto>>> GetAllShippers()
        {
            var shippersDto = await _shipperService.GetAllShippersAsync();
            return Ok(shippersDto);
        }

        [HttpPost]
        public async Task<ActionResult<int>> AddShipper(WriteShipperDto shipperDto)
        {
            var shipperId = await _shipperService.AddShipperAsync(shipperDto);
            return CreatedAtAction(nameof(GetShipperById), new { id = shipperId },
shipperId);
        }

        [HttpPut("{id}")]
        public async Task<IActionResult> UpdateShipper(int id, WriteShipperDto
shipperDto)
        {
            await _shipperService.UpdateShipperAsync(id, shipperDto);
            return NoContent();
        }

        [HttpDelete("{id}")]
    }
}
```

```

        public async Task<IActionResult> RemoveShipper(int id)
    {
        await _shipperService.RemoveShipperAsync(id);
        return NoContent();
    }
}

```

## SupplierController

```

using Microsoft.AspNetCore.Mvc;
using OnBudget.BL.DTOs.SupplierDtos;
using OnBudget.BL.Services.SupplierService;

namespace OnBudget.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class SupplierController : ControllerBase
    {
        private readonly ISupplierService _supplierService;

        public SupplierController(ISupplierService supplierService)
        {
            _supplierService = supplierService;
        }

        [HttpGet("{Handle}")]
        public async Task<ActionResult<ReadSupplierDto>> GetByUsernameAsync(string Handle)
        {
            var supplierDto = await _supplierService.GetByUsernameAsync(Handle);
            if (supplierDto == null)
            {
                return NotFound();
            }
            return Ok(supplierDto);
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<ReadSupplierDto>>>
GetAllSuppliers()
        {
            var suppliersDto = await _supplierService.GetAllSuppliersAsync();
            return Ok(suppliersDto);
        }

        [HttpPost]
        public async Task<ActionResult<string>> AddSupplier(WriteSupplierDto supplierDto)
        {

```

```

        var supplierHandle = await
    _supplierService.AddSupplierAsync(supplierDto);
        return Ok(supplierHandle);
    }

    [HttpPut("{Handle}")]
    public async Task<IActionResult> UpdateSupplier(string Handle,
WriteSupplierDto supplierDto)
    {
        await _supplierService.UpdateSupplierAsync(Handle, supplierDto);
        return NoContent();
    }

    [HttpDelete("{Handle}")]
    public async Task<IActionResult> RemoveSupplier(string Handle)
    {
        await _supplierService.RemoveSupplierAsync(Handle);
        return NoContent();
    }
}

```

## 5.4 AI ,Machine Learning and Deep learning

### 5.4.1 Size recommendation:

The model focuses on building and evaluating a neural network model to predict clothing size categories based on survey data. It preprocesses the data by merging datasets, handling missing values, and removing outliers. The data is then balanced using SMOTE, scaled, and the target variable is encoded. Finally, the dataset is split into training and testing sets, and a neural network model is trained and tested to predict clothing sizes, with its performance evaluated on the test set and with external data to ensure robust validation.

## Dataset:

**The dataset is Kaggle dataset, which contains 120,000 samples across 4 columns:**

**The Dataset consists of 4 columns:**

- 1. Weight (in kgs)**
- 2. Age**
- 3. Height (in cm)**
- 4. Size**

**The output parameter is size and input parameters are Weight, Age, and Height.**

## Code Implementation:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import numpy as np

warnings.filterwarnings("ignore")
# # Load the dataset

# df1 = pd.read_csv("/content/final_test.csv")
# df2 = pd.read_csv("/content/Clothing Size Survey.csv")
# # Concatenate the two DataFrames along the rows axis
# (vertically stacking them)
# df = pd.concat([df1, df2])
# df.to_csv('df.csv')

df = pd.read_csv("/content/final_test_2.csv")

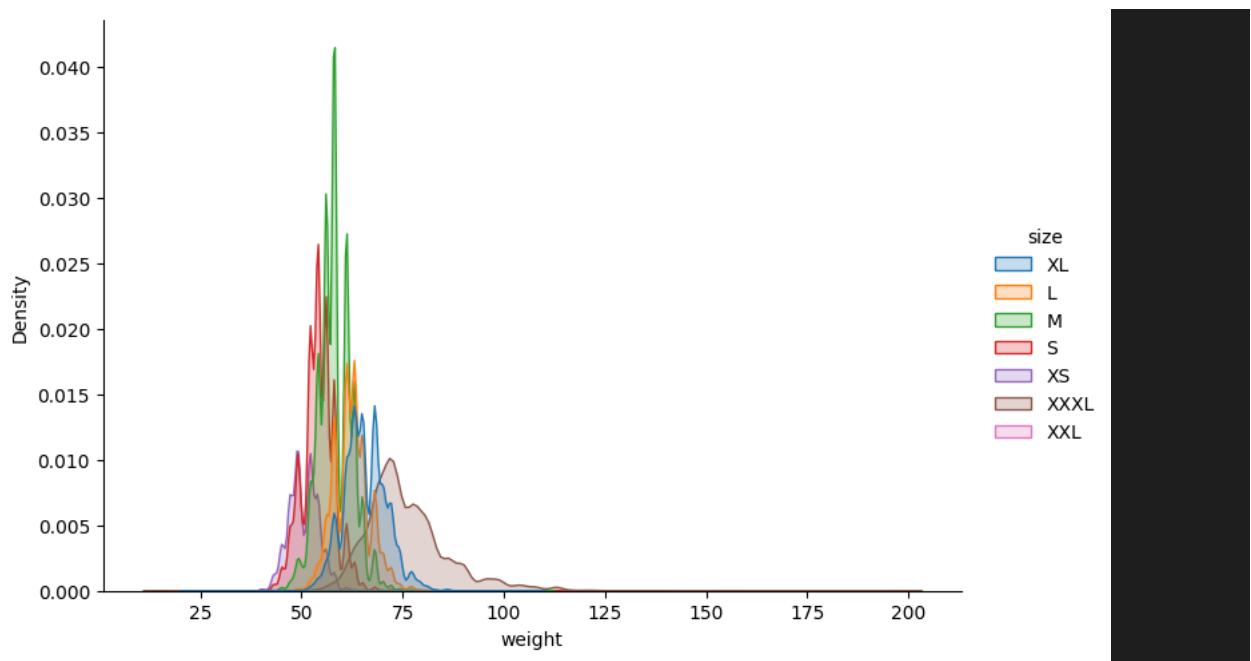
df
```

weight	age	height	size	
0	62	28.0	172.72	XL
1	59	36.0	167.64	L
2	61	34.0	165.10	M
3	65	27.0	175.26	L
4	62	45.0	172.72	M
...	...	...	...	...
119914	55	41.0	158.00	M
119915	70	52.0	165.00	XL
119916	92	61.0	178.00	XXL
119917	15	5.0	100.00	XXXL
119918	47	22.0	160.00	M

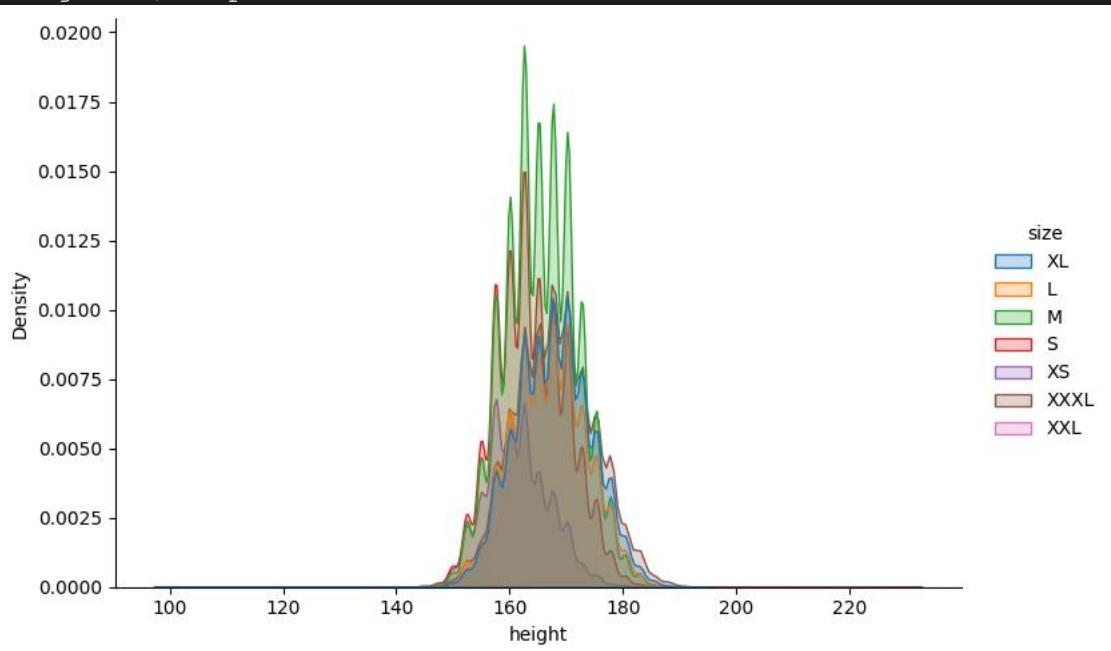
119919 rows × 4 columns

## EDA ,Visualization and Data cleaning

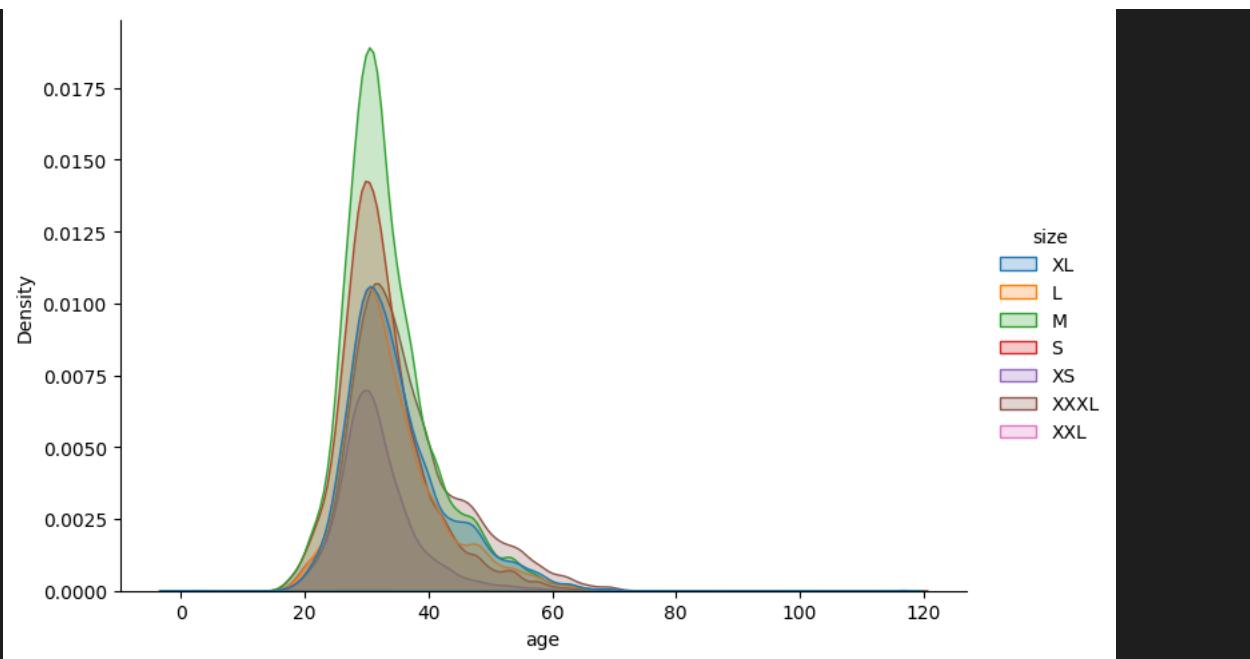
```
df['size'] = df['size'].replace('XXS', 'XS')
sns.displot(df, x="weight", hue="size", kind="kde", fill=True ,
height=5, aspect=1.5)
```



```
sns.displot(df, x="height", hue="size", kind="kde", fill=True ,  
height=5, aspect=1.5)
```



```
sns.displot(df, x="age", hue="size", kind="kde", fill=True ,  
height=5, aspect=1.5)
```



```
df['weight'].describe()
```

```
count    119919.000000
```

```
mean     61.772213
```

```
std      9.980826
```

```
min     15.000000
```

```
25%     55.000000
```

```
50%     61.000000
```

```
75%     67.000000
```

```
max     199.000000
```

```
Name: weight, dtype: float64
```

```
df.head(20)
```

```
df.isnull().sum()
```

```
weight    0
```

```
age     257
```

```
height   330
```

```
size     0
```

```
dtype: int64
```

```
df.describe()
```

```
    weight    age    height
```

```
count    119919.000000  119662.000000  119589.000000
```

```
mean   61.772213  34.012243  165.809632
```

```
std    9.980826  8.160997  6.766011
```

```

min   15.000000 0.000000 100.000000
25%  55.000000 29.000000 160.020000
50%  61.000000 32.000000 165.100000
75%  67.000000 37.000000 170.180000
max   199.000000      117.000000      230.000000

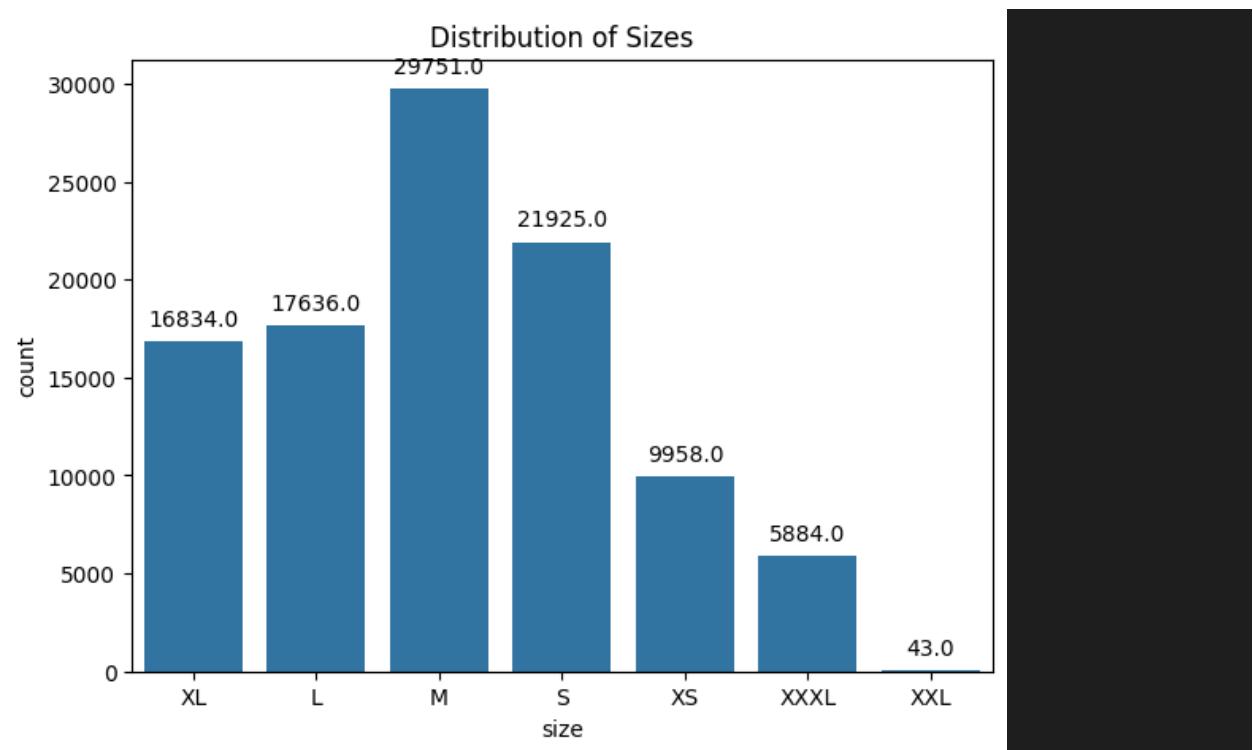
# drop where size is XXXL and weight is less than 80
df   = df.drop(df[(df['size'] == 'XXXL') & (df['weight'] < 80)].index)
# drop where size is XXL and weight is less than 70
df   = df.drop(df[(df['size'] == 'XXL') & (df['weight'] < 70)].index)
# drop where size is XL and weight is less than 60
df = df.drop(df[(df['size'] == 'XL') & (df['weight'] < 60)].index)
# drop where size is xxs and weight is greater than 60
df   = df.drop(df[(df['size'] == 'XXS') & (df['weight'] > 60)].index)
# drop where size is xs and weight is greater than 70
df = df.drop(df[(df['size'] == 'XS') & (df['weight'] > 70)].index)
# drop where size is s and weight is greater than 80
df = df.drop(df[(df['size'] == 'S') & (df['weight'] > 80)].index)

# Distribution of the target variable 'size' with count
# annotations
plt.figure(figsize=(7, 5))
sns.countplot(x='size', data=df)
plt.title('Distribution of Sizes')

# Annotate each bar with its count
for p in plt.gca().patches:
    plt.gca().annotate(f'{p.get_height()}', (p.get_x() +
p.get_width() / 2., p.get_height()),
                       ha='center', va='center', xytext=(0, 10),
textcoords='offset points')

plt.show()

```



```
# generate artificial data to balance the dataset for the target
variable 'size'
from imblearn.over_sampling import SMOTE
from collections import Counter
# weight age height size
sm = SMOTE(random_state=42, sampling_strategy='all')
# remove null values
df = df.dropna()
# drop duplicates
df = df.drop_duplicates()

# resample the dataset with max value of 10000
X_res, y_res = sm.fit_resample(df[['weight', 'age', 'height']],
df['size'])

# add the target variable 'size' to the resampled dataset
X_res['size'] = y_res

print('Resampled dataset shape %s' % Counter(y_res))

# Distribution of the target variable 'size' with count
# annotations
plt.figure(figsize=(6, 4))
```

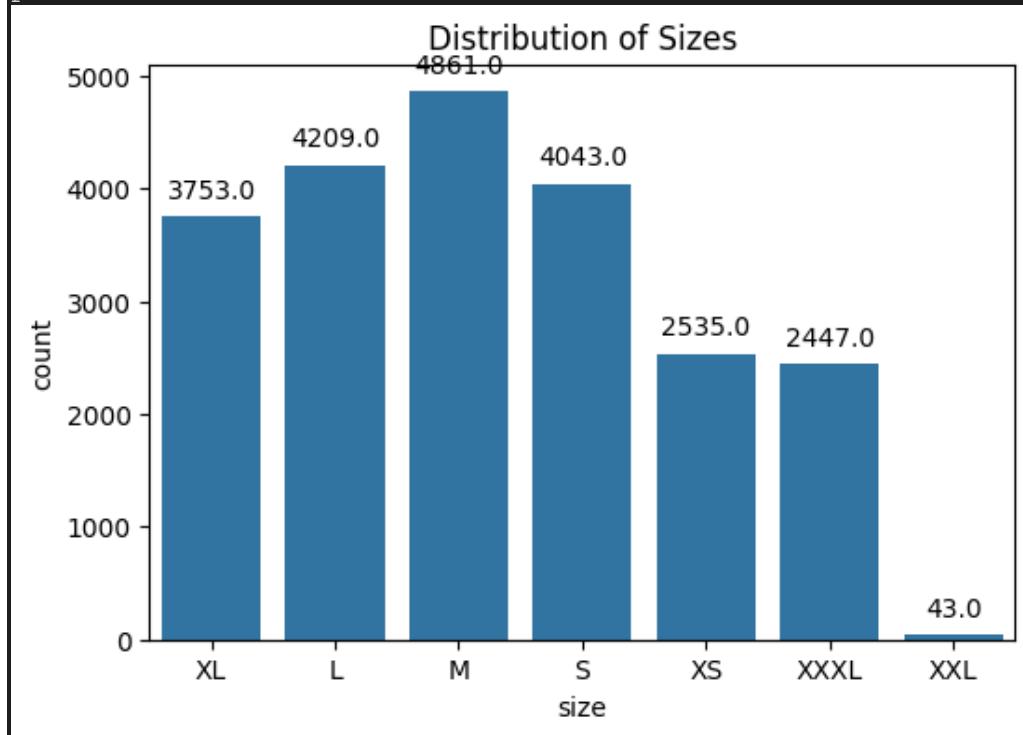
```

sns.countplot(x='size', data=df)
plt.title('Distribution of Sizes')

# Annotate each bar with its count
for p in plt.gca().patches:
    plt.gca().annotate(f'{p.get_height()}', (p.get_x() +
p.get_width() / 2., p.get_height()),
                       ha='center', va='center', xytext=(0, 10),
textcoords='offset points')

plt.show()

```



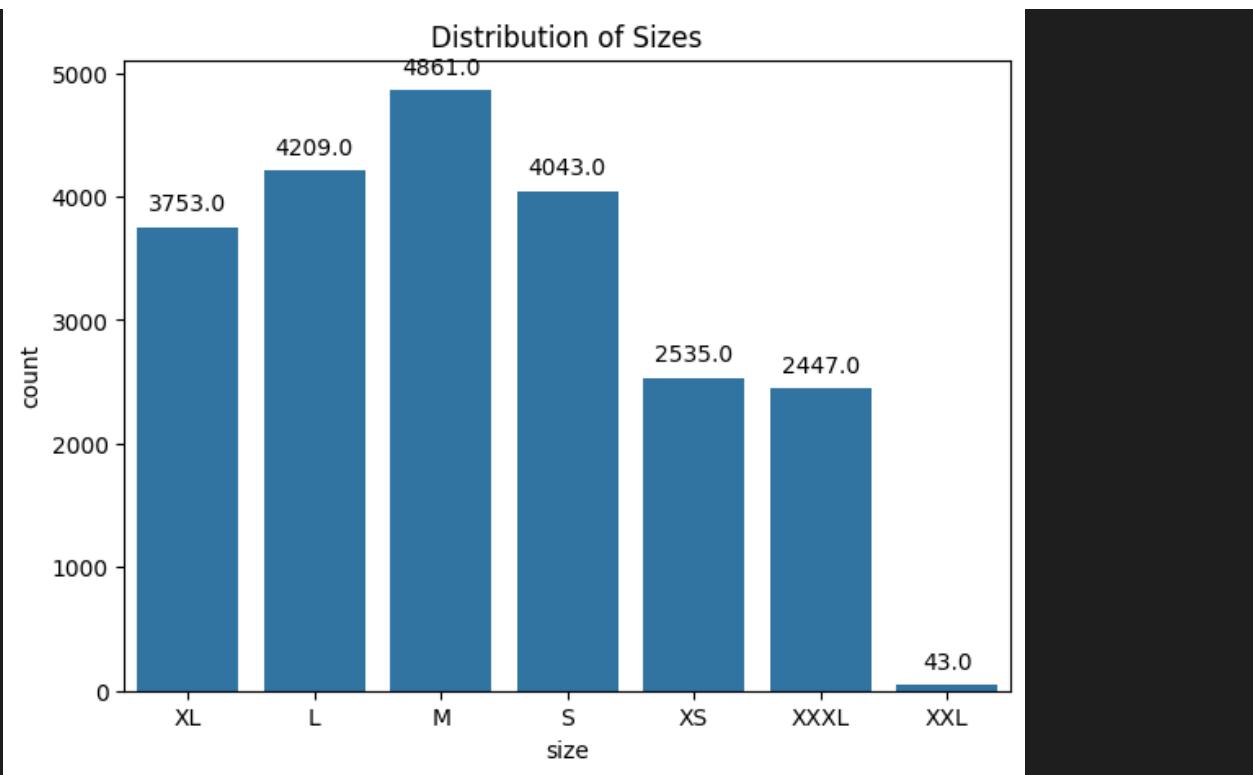
```

# Distribution of the target variable 'size' with count
# annotations
plt.figure(figsize=(7, 5))
sns.countplot(x='size', data=df)
plt.title('Distribution of Sizes')

# Annotate each bar with its count
for p in plt.gca().patches:
    plt.gca().annotate(f'{p.get_height()}', (p.get_x() +
p.get_width() / 2., p.get_height()),
                       ha='center', va='center', xytext=(0, 10),
textcoords='offset points')

plt.show()

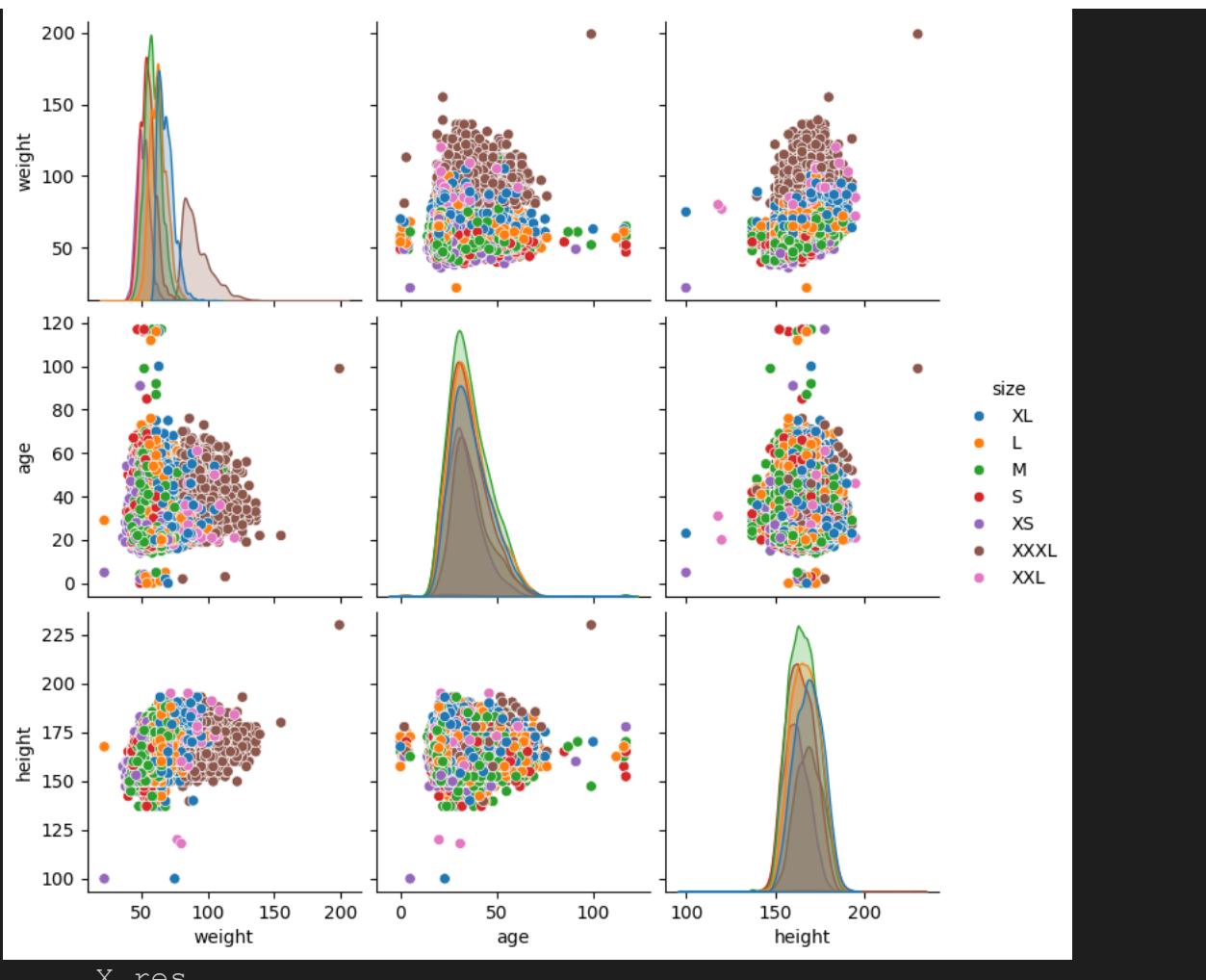
```



```
df.describe()
weight      age   height
count    21891.000000  21891.000000  21891.000000
mean   62.954273 35.527112 165.836209
std    13.416012 10.540272  8.201134
min    22.000000  0.000000 100.000000
25%   54.000000 28.000000 160.020000
50%   61.000000 34.000000 165.100000
75%   68.000000 42.000000 172.720000
max   199.000000       117.000000       230.000000

#distribution of each column

# sns.pairplot(df[['weight', 'age', 'height', 'size']])
sns.pairplot(df[['weight', 'age', 'height', 'size']],
hue='size')
plt.show()
```



X\_res

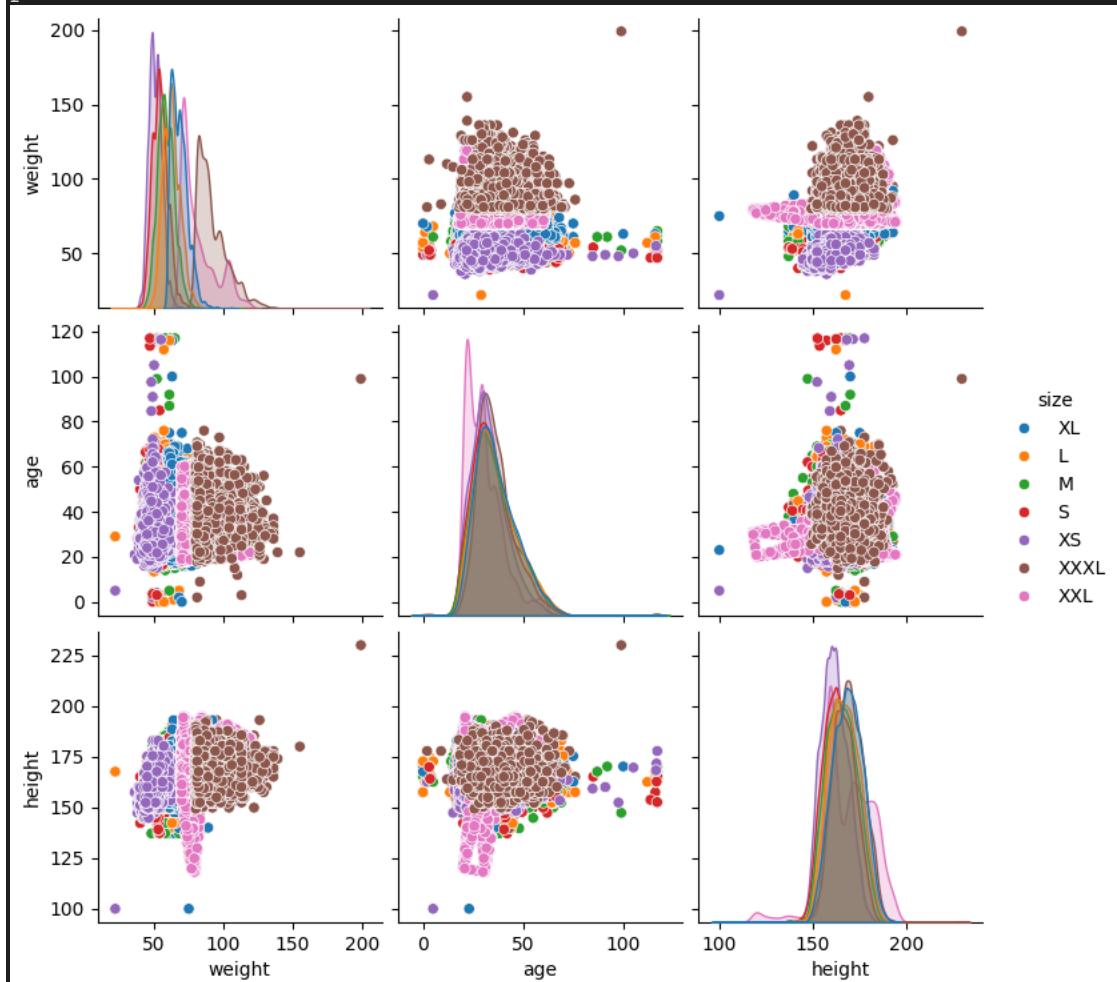
weight	age	height	size	
0	62	28.000000	172.72000	XL
1	59	36.000000	167.64000	L
2	61	34.000000	165.10000	M
3	65	27.000000	175.26000	L
4	62	45.000000	172.72000	M
...	...	...	...	...
34022	97	41.626457	167.64000	XXXL
34023	106	19.287327	170.90981	XXXL

weight	age	height	size
34024	81	42.559492	165.10000 XXXL
34025	89	40.331998	165.10000 XXXL
34026	82	53.674407	167.64000 XXXL

34027 rows × 4 columns

```
#distribution of each column
# sns.pairplot(X_res[['weight', 'age', 'height', 'size']])
sns.pairplot(X_res[['weight', 'age', 'height', 'size']],
hue='size')
```

plt.show()



df = X\_res

```

len(df)

34027

# Check for missing values
missing_values = df.isnull().sum()
print(missing_values)
weight    0
age       0
height   0
size      0
dtype: int64

#replace missing values with mean
df['age'].fillna(df['age'].mean(), inplace=True)
df['height'].fillna(df['height'].mean(), inplace=True)

df.dtypes
Os
weight    int64
age      float64
height   float64
size     object
dtype: object

# Identify object columns
object_columns = df.select_dtypes(include=['object']).columns

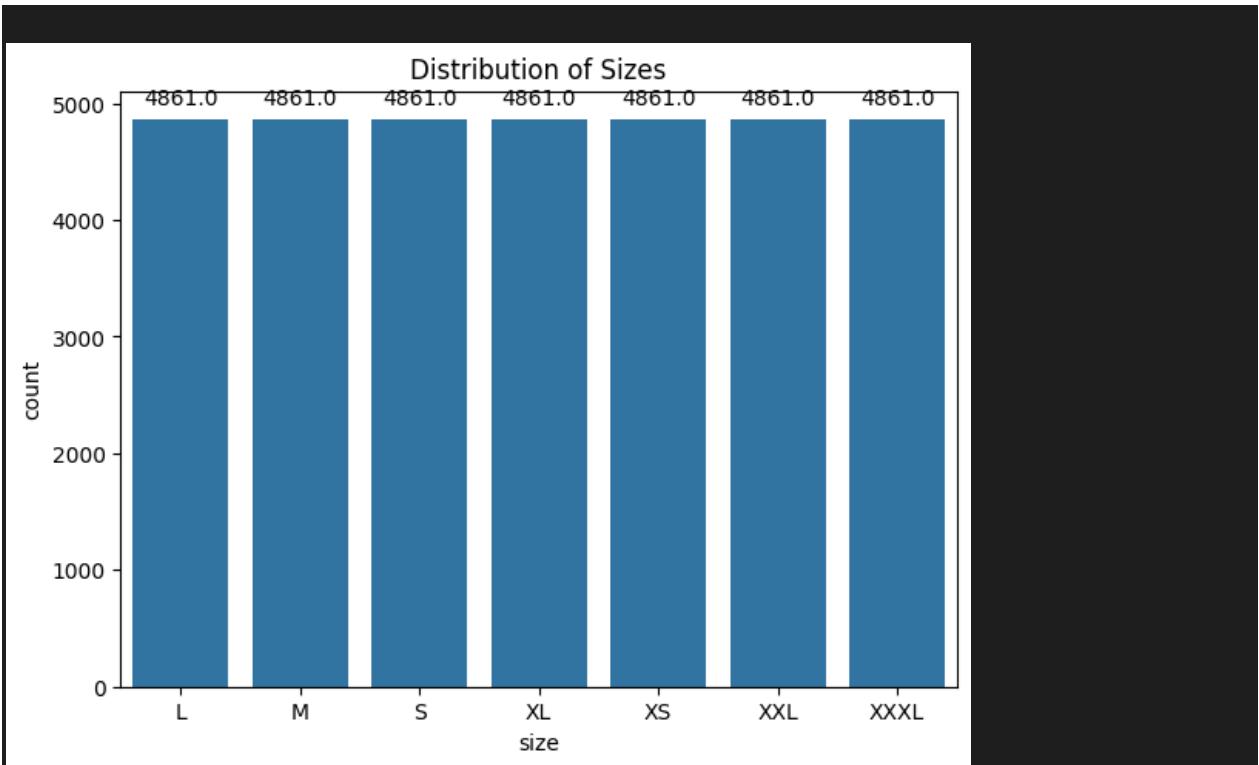
# Convert object columns to category
df[object_columns] = df[object_columns].astype('category')

# Distribution of the target variable 'size' with count
# annotations
plt.figure(figsize=(7,5))
sns.countplot(x='size', data=df)
plt.title('Distribution of Sizes')

# Annotate each bar with its count
for p in plt.gca().patches:
    plt.gca().annotate(f'{p.get_height()}', (p.get_x() +
p.get_width() / 2., p.get_height()),
                       ha='center', va='center', xytext=(0, 10),
textcoords='offset points')

plt.show()

```



```
df['size'] = df['size'].astype('object')

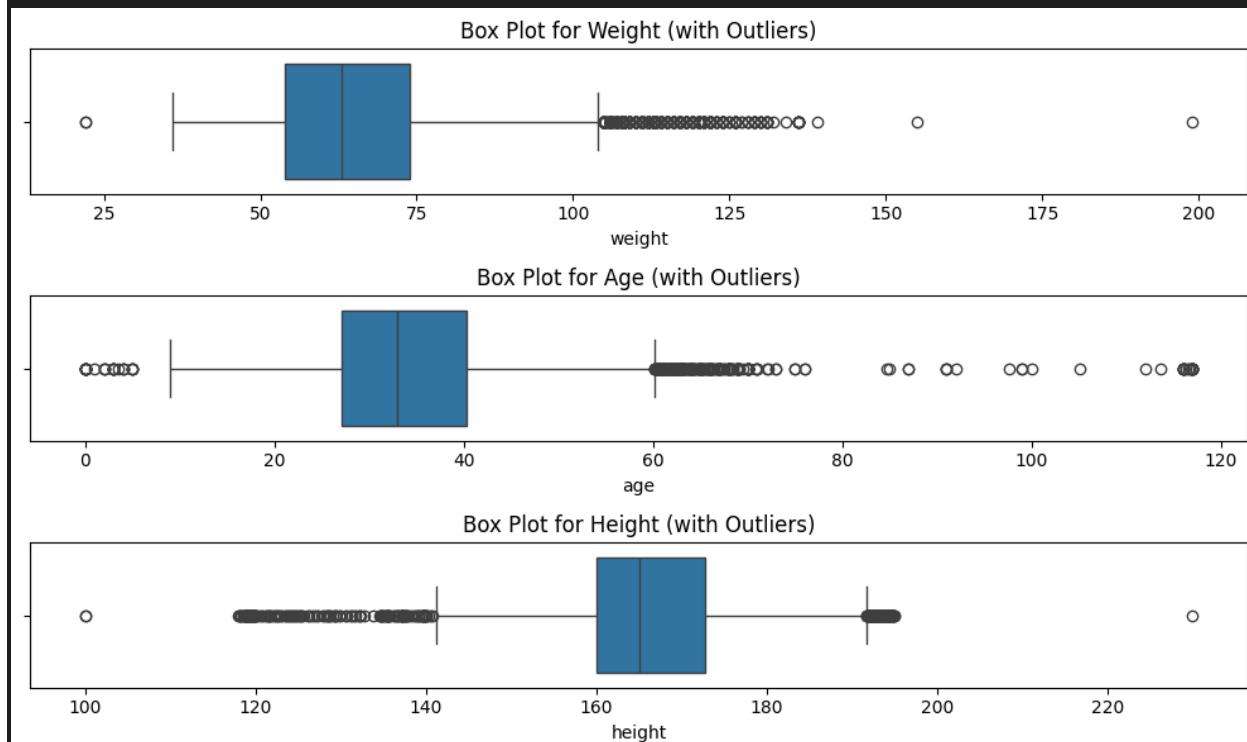
# Check for missing values
missing_values = df.isnull().sum()
missing_values

weight    0
age      0
height   0
size     0
dtype: int64

# Visualize outliers using box plots for all numerical features
plt.figure(figsize=(10, 6))
# Box plot for 'weight'
plt.subplot(3, 1, 1)
sns.boxplot(x=df['weight'])
plt.title('Box Plot for Weight (with Outliers)')

# Box plot for 'age'
plt.subplot(3, 1, 2)
sns.boxplot(x=df['age'])
plt.title('Box Plot for Age (with Outliers)')
```

```
# Box plot for 'height'
plt.subplot(3, 1, 3)
sns.boxplot(x=df['height'])
plt.title('Box Plot for Height (with Outliers)')
plt.tight_layout()
plt.show()
```



```
# Dealing with the outliers in the 'age' column

# Get the first quartile (Q1) and third quartile (Q3)
Q1 = df['age'].quantile(0.25)
Q3 = df['age'].quantile(0.75)

# Calculate the Interquartile Range (IQR)
IQR = Q3 - Q1

# Calculate the upper and lower bounds
upper_bound = Q3 + 1.5 * IQR
lower_bound = Q1 - 1.5 * IQR

# Replace outliers with upper and lower bounds
df['age'] = df['age'].apply(lambda x: upper_bound if x >
upper_bound else (lower_bound if x < lower_bound else x))
```

```

# Visualize outliers using box plots for all numerical features
plt.figure(figsize=(10, 6))

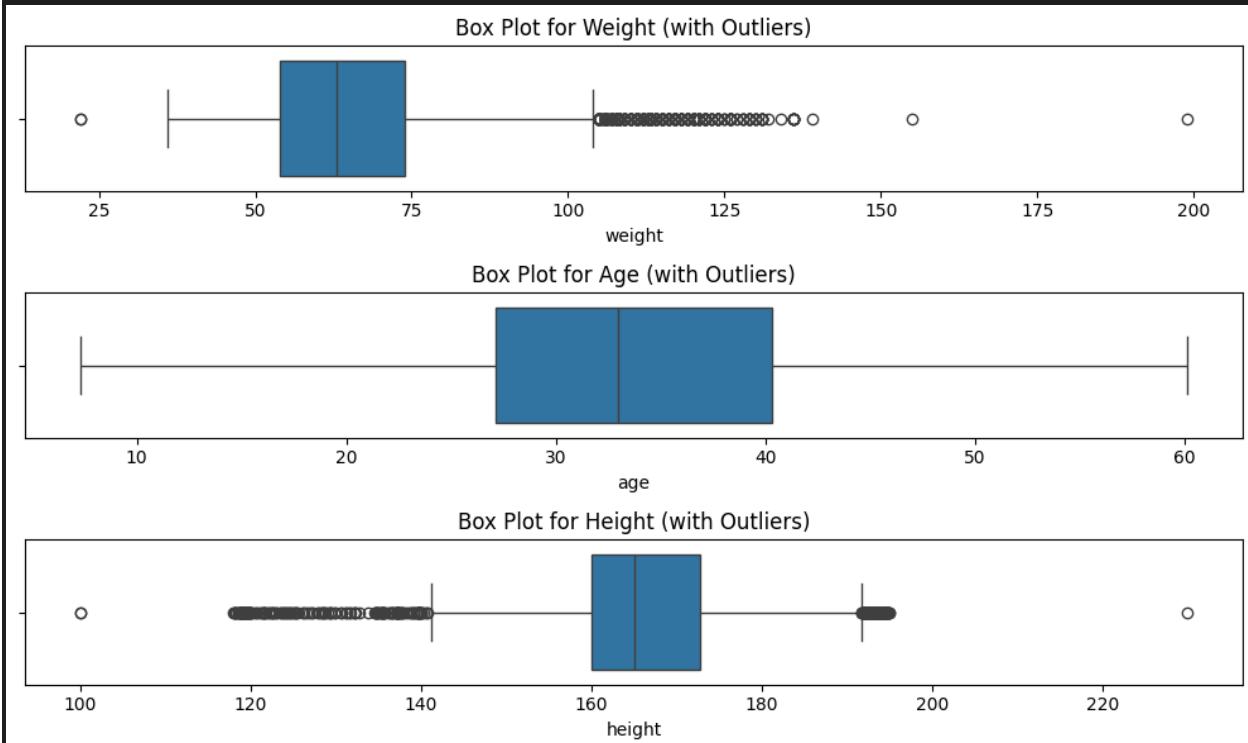
# Box plot for 'weight'
plt.subplot(3, 1, 1)
sns.boxplot(x=df['weight'])
plt.title('Box Plot for Weight (with Outliers)')

# Box plot for 'age'
plt.subplot(3, 1, 2)
sns.boxplot(x=df['age'])
plt.title('Box Plot for Age (with Outliers)')

# Box plot for 'height'
plt.subplot(3, 1, 3)
sns.boxplot(x=df['height'])
plt.title('Box Plot for Height (with Outliers)')

plt.tight_layout()
plt.show()

```



```

df['weight'].describe()
count    34027.000000

```

```

mean    66.690452
std     16.050005
min     22.000000
25%    54.000000
50%    63.000000
75%    74.000000
max    199.000000
Name: weight, dtype: float64

# Calculate BMI
df['bmi'] = df['weight'] / ((df['height'] / 100) ** 2)

# Display the DataFrame with the new 'bmi' column
print("DataFrame with BMI Column:\n", df.head())

# remove bmi outliers using IQR
# Get the first quartile (Q1) and third quartile (Q3)
Q1 = df['bmi'].quantile(0.25)
Q3 = df['bmi'].quantile(0.75)

# Calculate the Interquartile Range (IQR)
IQR = Q3 - Q1

# Calculate the upper and lower bounds
upper_bound = Q3 + 1.5 * IQR
lower_bound = Q1 - 1.5 * IQR

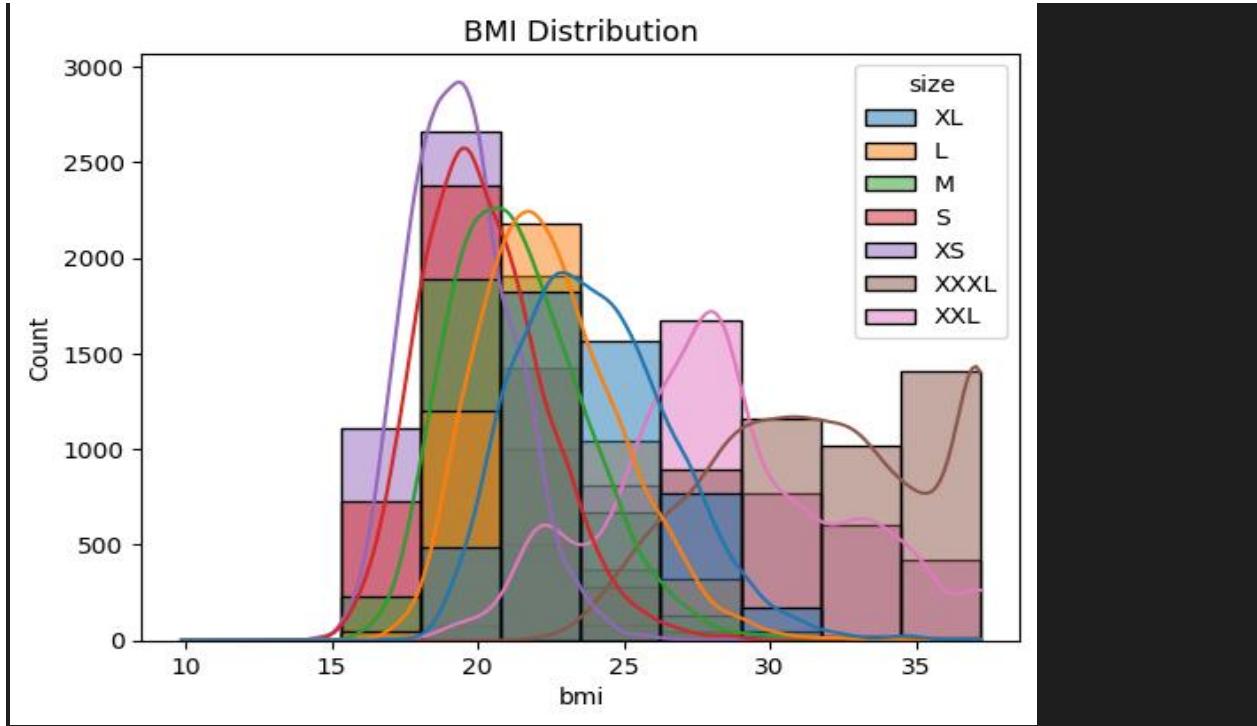
# Replace outliers with upper and lower bounds
df['bmi'] = df['bmi'].apply(lambda x: upper_bound if x >
upper_bound else (lower_bound if x < lower_bound else x))

# Visualize bmi histogram with size as hue
sns.histplot(data=df, x='bmi', hue='size', kde=True, bins=10)
plt.title('BMI Distribution')
plt.show()

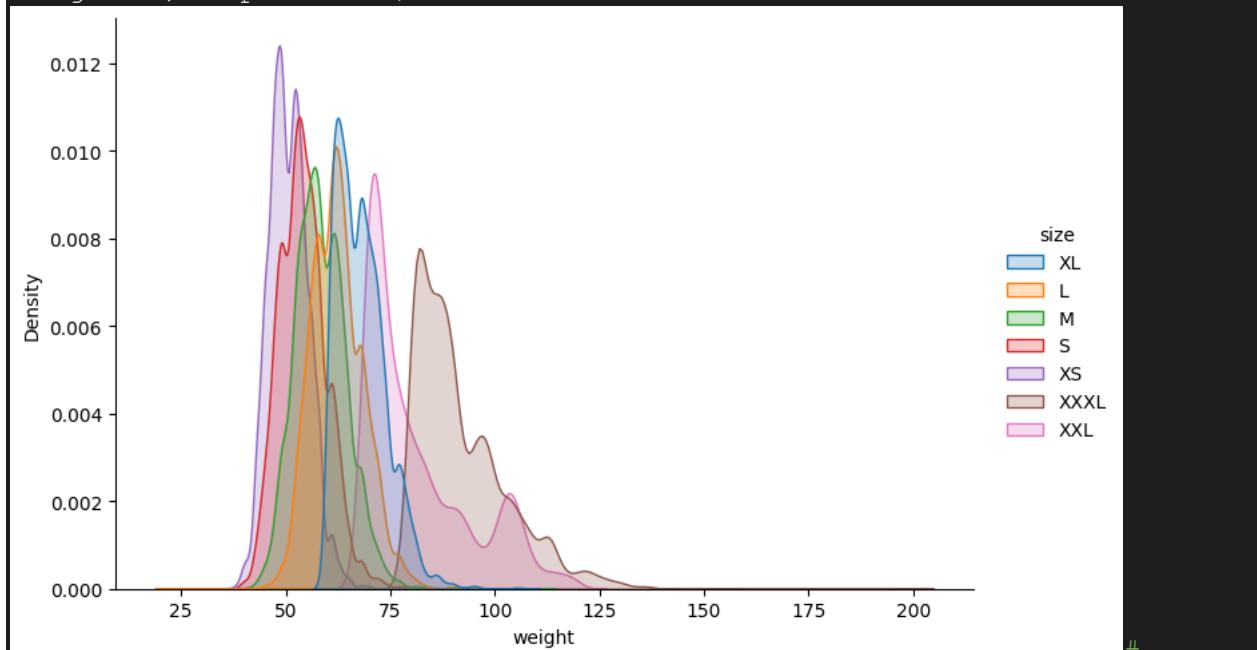
# df_new = df.copy()
# # if bmi is less than 18.5, then size is XS
# df_new.loc[df_new['bmi'] < 18.5, 'size'] = 'XS'
DataFrame with BMI Column:
   weight   age  height size      bmi
0      62  28.0   172.72   XL  20.782914
1      59  36.0   167.64    L  20.994073
2      61  34.0   165.10    M  22.378743
3      65  27.0   175.26    L  21.161563

```

```
4       62    45.0   172.72      M   20.782914
```



```
# Visualize weight histogram with size as hue with large figure
size
sns.displot(df, x="weight", hue="size", kind="kde", fill=True ,
height=5, aspect=1.5)
```

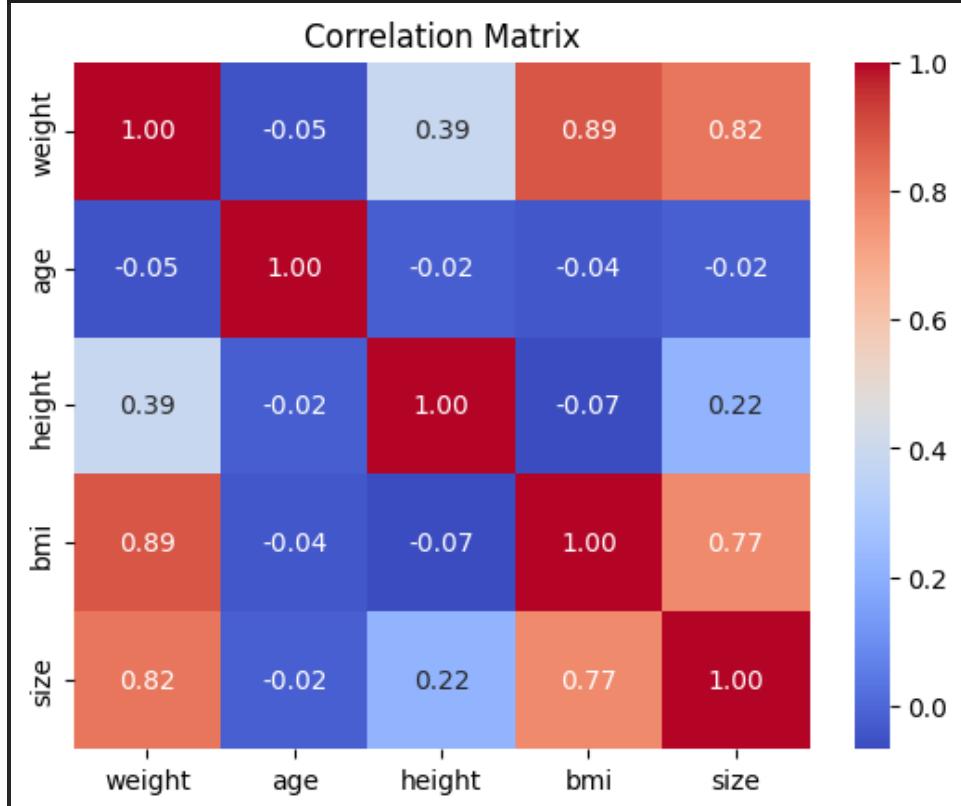


```
Correlation matrix for bmi
correlation_matrix = df.corr()
```

```

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Matrix')

```



## Normalization and Encoding

```

#normalization
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
# df[['weight', 'age',
'height','neck_circumference','waist_circumference','bmi',
'hip_circumference', 'whr', 'shoulder_circumference',
'shoulder_to_waist_ratio', 'torso_length', 'leg_length']] =
scaler.fit_transform(df[['weight', 'age',
'height','neck_circumference','bmi', 'waist_circumference',
'hip_circumference', 'whr', 'shoulder_circumference',
'shoulder_to_waist_ratio', 'torso_length', 'leg_length']])
df[['weight', 'age', 'height','bmi']] =
scaler.fit_transform(df[['weight', 'age', 'height', 'bmi']])

# Encoding the target variable 'size'
size_mapping = {'XXS': 1, 'S': 2, 'M': 3, 'L': 4, 'XL': 5,
'XXL': 6, 'XXXL': 7}

```

```

df['size'] = df['size'].map(size_mapping)

# df_encoded = df[['weight', 'age',
# 'height', 'neck_circumference', 'bmi', 'waist_circumference',
# 'hip_circumference', 'whr', 'shoulder_circumference',
# 'shoulder_to_waist_ratio', 'torso_length',
# 'leg_length', 'size']]
df_encoded = df[['weight', 'age', 'height', 'bmi', 'size']]
df=df_encoded
df

```

	<b>weight</b>	<b>age</b>	<b>height</b>	<b>bmi</b>	<b>size</b>	
<b>0</b>	0.225989	0.391721	0.559385	0.399525	5.0	
<b>1</b>	0.209040	0.543074	0.520308	0.407250	4.0	
<b>2</b>	0.220339	0.505235	0.500769	0.457905	3.0	
<b>3</b>	0.242938	0.372802	0.578923	0.413377	4.0	
<b>4</b>	0.225989	0.713345	0.559385	0.399525	3.0	
...	...	...	...	...	...	...
<b>34022</b>	0.423729	0.649521	0.520308	0.901903	7.0	
<b>34023</b>	0.474576	0.226885	0.545460	0.966765	7.0	
<b>34024</b>	0.333333	0.667173	0.500769	0.726320	7.0	
<b>34025</b>	0.378531	0.625031	0.500769	0.833687	7.0	
<b>34026</b>	0.338983	0.877457	0.520308	0.706645	7.0	

34027 rows × 5 columns

```

columns = df.columns.to_list()
columns.append(columns.pop(columns.index("size")))

# Distribution of the target variable 'size' with count
# annotations
plt.figure(figsize=(7, 5))
sns.countplot(x='size', data=df)
plt.title('Distribution of Sizes')

# Annotate each bar with its count

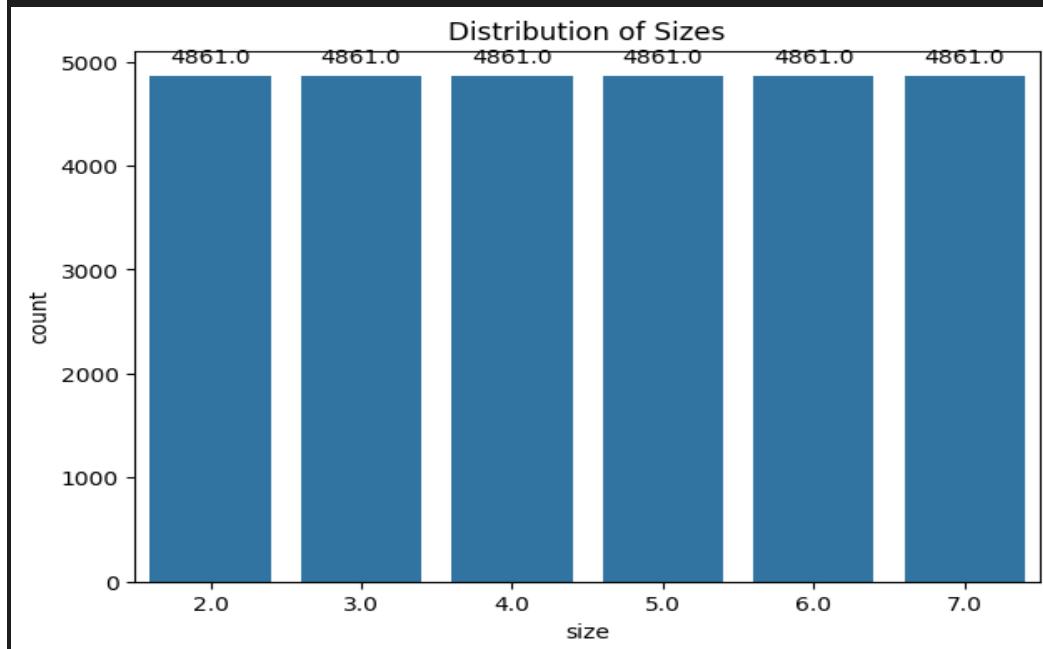
```

```

for p in plt.gca().patches:
    plt.gca().annotate(f'{p.get_height()}', (p.get_x() +
p.get_width() / 2., p.get_height()),
                       ha='center', va='center', xytext=(0, 10),
textcoords='offset points')

plt.show()

```



## Splitting the data

```

from sklearn.model_selection import train_test_split

# Splitting the dataset into features (X) and target variable (y)
X = df.drop(columns=['size']) # Features
y = df['size'] # Target variable

# Splitting the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Displaying the shapes of the train and test sets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)

```

```
print("y_test shape:", y_test.shape)
```

```
x_train shape: (27221, 4)
```

```
x_test shape: (6806, 4)
```

```
y_train shape: (27221,)
```

```
y_test shape: (6806,)
```

```
df
```

weight	age	height	bmi	size	
0	0.225989	0.391721	0.559385	0.399525	5.0
1	0.209040	0.543074	0.520308	0.407250	4.0
2	0.220339	0.505235	0.500769	0.457905	3.0
3	0.242938	0.372802	0.578923	0.413377	4.0
4	0.225989	0.713345	0.559385	0.399525	3.0
...	...	...	...	...	...
34022	0.423729	0.649521	0.520308	0.901903	7.0
34023	0.474576	0.226885	0.545460	0.966765	7.0
34024	0.333333	0.667173	0.500769	0.726320	7.0
34025	0.378531	0.625031	0.500769	0.833687	7.0
34026	0.338983	0.877457	0.520308	0.706645	7.0

```
34027 rows × 5 columns
```

```
# Check for NaN values in y_train
```

```
nan_indices = y_train.index[y_train.isna()]
```

```
print("Indices of NaN values in y_train:", nan_indices)
```

```
# Remove rows with NaN values from x_train and y_train
```

```
x_train = x_train.drop(index=nan_indices)
```

```
y_train = y_train.drop(index=nan_indices)
```

# TensorFlow Model

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam

# Define the model architecture
model = Sequential([
    Dense(64, activation='relu',
input_shape=(X_train.shape[1],)),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(1) # Output layer with one neuron for regression
])

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001),
loss='mean_squared_error', metrics=['mae'])

# Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=32,
validation_split=0.2, verbose=1)

# Evaluate the model on the test set
loss, mae = model.evaluate(X_test, y_test)

print("Test Loss:", loss)
print("Test MAE:", mae)
Epoch 1/50
584/584 [=====] - 3s 4ms/step - loss: 2.8001 - mae: 1.2355 - val_loss: 0.9364 - val_mae: 0.8128
Epoch 2/50
584/584 [=====] - 2s 4ms/step - loss: 1.1898 - mae: 0.8917 - val_loss: 0.8333 - val_mae: 0.7655
Epoch 3/50
584/584 [=====] - 2s 4ms/step - loss: 1.1039 - mae: 0.8544 - val_loss: 0.7796 - val_mae: 0.7351
Epoch 4/50
584/584 [=====] - 2s 4ms/step - loss: 1.0794 - mae: 0.8464 - val_loss: 0.7486 - val_mae: 0.7167
Epoch 5/50
584/584 [=====] - 3s 5ms/step - loss: 1.0323 - mae: 0.8262 - val_loss: 0.7371 - val_mae: 0.7093
```

```
Epoch 6/50
584/584 [=====] - 4s 6ms/step - loss: 1.0144 - mae: 0.8196 - val_loss: 0.7161 - val_mae: 0.6945
Epoch 7/50
584/584 [=====] - 2s 4ms/step - loss: 0.9692 - mae: 0.8018 - val_loss: 0.6952 - val_mae: 0.6844
Epoch 8/50
584/584 [=====] - 3s 5ms/step - loss: 0.9245 - mae: 0.7813 - val_loss: 0.6833 - val_mae: 0.6734
Epoch 9/50
584/584 [=====] - 3s 5ms/step - loss: 0.9074 - mae: 0.7709 - val_loss: 0.6707 - val_mae: 0.6617
Epoch 10/50
584/584 [=====] - 3s 4ms/step - loss: 0.8756 - mae: 0.7567 - val_loss: 0.6644 - val_mae: 0.6595
Epoch 11/50
584/584 [=====] - 2s 4ms/step - loss: 0.8594 - mae: 0.7518 - val_loss: 0.6532 - val_mae: 0.6464
Epoch 12/50
584/584 [=====] - 2s 4ms/step - loss: 0.8362 - mae: 0.7396 - val_loss: 0.6462 - val_mae: 0.6563
Epoch 13/50
584/584 [=====] - 2s 4ms/step - loss: 0.8283 - mae: 0.7378 - val_loss: 0.6533 - val_mae: 0.6619
Epoch 14/50
584/584 [=====] - 2s 4ms/step - loss: 0.8204 - mae: 0.7345 - val_loss: 0.6455 - val_mae: 0.6492
Epoch 15/50
584/584 [=====] - 3s 6ms/step - loss: 0.8070 - mae: 0.7294 - val_loss: 0.6343 - val_mae: 0.6490
Epoch 16/50
584/584 [=====] - 2s 4ms/step - loss: 0.8022 - mae: 0.7269 - val_loss: 0.6261 - val_mae: 0.6380
Epoch 17/50
584/584 [=====] - 2s 4ms/step - loss: 0.7809 - mae: 0.7151 - val_loss: 0.6266 - val_mae: 0.6372
Epoch 18/50
584/584 [=====] - 2s 4ms/step - loss: 0.7872 - mae: 0.7200 - val_loss: 0.6310 - val_mae: 0.6443
Epoch 19/50
584/584 [=====] - 2s 4ms/step - loss: 0.7774 - mae: 0.7146 - val_loss: 0.6324 - val_mae: 0.6384
Epoch 20/50
584/584 [=====] - 3s 4ms/step - loss: 0.7748 - mae: 0.7115 - val_loss: 0.6331 - val_mae: 0.6438
Epoch 21/50
```

```
584/584 [=====] - 3s 5ms/step - loss:  
0.7718 - mae: 0.7116 - val_loss: 0.6386 - val_mae: 0.6496  
Epoch 22/50  
584/584 [=====] - 2s 4ms/step - loss:  
0.7626 - mae: 0.7059 - val_loss: 0.6428 - val_mae: 0.6593  
Epoch 23/50  
584/584 [=====] - 2s 4ms/step - loss:  
0.7640 - mae: 0.7066 - val_loss: 0.6334 - val_mae: 0.6453  
Epoch 24/50  
584/584 [=====] - 2s 4ms/step - loss:  
0.7497 - mae: 0.6997 - val_loss: 0.6702 - val_mae: 0.6516  
Epoch 25/50  
584/584 [=====] - 2s 4ms/step - loss:  
0.7542 - mae: 0.7033 - val_loss: 0.6203 - val_mae: 0.6333  
Epoch 26/50  
584/584 [=====] - 3s 5ms/step - loss:  
0.7478 - mae: 0.7003 - val_loss: 0.6366 - val_mae: 0.6571  
Epoch 27/50  
584/584 [=====] - 2s 4ms/step - loss:  
0.7468 - mae: 0.7001 - val_loss: 0.6270 - val_mae: 0.6407  
Epoch 28/50  
584/584 [=====] - 2s 4ms/step - loss:  
0.7381 - mae: 0.6942 - val_loss: 0.6391 - val_mae: 0.6499  
Epoch 29/50  
584/584 [=====] - 2s 4ms/step - loss:  
0.7403 - mae: 0.6958 - val_loss: 0.6387 - val_mae: 0.6528  
Epoch 30/50  
584/584 [=====] - 2s 4ms/step - loss:  
0.7411 - mae: 0.6952 - val_loss: 0.6408 - val_mae: 0.6535  
Epoch 31/50  
584/584 [=====] - 3s 5ms/step - loss:  
0.7271 - mae: 0.6914 - val_loss: 0.6328 - val_mae: 0.6423  
Epoch 32/50  
584/584 [=====] - 3s 5ms/step - loss:  
0.7294 - mae: 0.6912 - val_loss: 0.6337 - val_mae: 0.6428  
Epoch 33/50  
584/584 [=====] - 2s 4ms/step - loss:  
0.7235 - mae: 0.6900 - val_loss: 0.6455 - val_mae: 0.6468  
Epoch 34/50  
584/584 [=====] - 2s 4ms/step - loss:  
0.7241 - mae: 0.6890 - val_loss: 0.6476 - val_mae: 0.6464  
Epoch 35/50  
584/584 [=====] - 2s 4ms/step - loss:  
0.7219 - mae: 0.6895 - val_loss: 0.6236 - val_mae: 0.6373  
Epoch 36/50  
584/584 [=====] - 2s 4ms/step - loss:  
0.7216 - mae: 0.6884 - val_loss: 0.6276 - val_mae: 0.6416
```

```

Epoch 37/50
584/584 [=====] - 3s 5ms/step - loss: 0.7178 - mae: 0.6866 - val_loss: 0.6380 - val_mae: 0.6451
Epoch 38/50
584/584 [=====] - 3s 5ms/step - loss: 0.7104 - mae: 0.6825 - val_loss: 0.6685 - val_mae: 0.6391
Epoch 39/50
584/584 [=====] - 3s 5ms/step - loss: 0.7166 - mae: 0.6837 - val_loss: 0.6824 - val_mae: 0.6537
Epoch 40/50
584/584 [=====] - 2s 4ms/step - loss: 0.7124 - mae: 0.6832 - val_loss: 0.6143 - val_mae: 0.6326
Epoch 41/50
584/584 [=====] - 2s 4ms/step - loss: 0.7189 - mae: 0.6862 - val_loss: 0.6420 - val_mae: 0.6523
Epoch 42/50
584/584 [=====] - 3s 5ms/step - loss: 0.7044 - mae: 0.6787 - val_loss: 0.6359 - val_mae: 0.6452
Epoch 43/50
584/584 [=====] - 2s 4ms/step - loss: 0.7041 - mae: 0.6797 - val_loss: 0.6772 - val_mae: 0.6422
Epoch 44/50
584/584 [=====] - 2s 4ms/step - loss: 0.7094 - mae: 0.6818 - val_loss: 0.6072 - val_mae: 0.6311
Epoch 45/50
584/584 [=====] - 2s 4ms/step - loss: 0.7021 - mae: 0.6772 - val_loss: 0.6203 - val_mae: 0.6426
Epoch 46/50
584/584 [=====] - 2s 4ms/step - loss: 0.7001 - mae: 0.6773 - val_loss: 0.6229 - val_mae: 0.6276
Epoch 47/50
584/584 [=====] - 2s 4ms/step - loss: 0.7044 - mae: 0.6777 - val_loss: 0.6134 - val_mae: 0.6258
Epoch 48/50
584/584 [=====] - 3s 5ms/step - loss: 0.7053 - mae: 0.6784 - val_loss: 0.6200 - val_mae: 0.6292
Epoch 49/50
584/584 [=====] - 2s 4ms/step - loss: 0.6988 - mae: 0.6752 - val_loss: 0.6277 - val_mae: 0.6250
Epoch 50/50
584/584 [=====] - 2s 4ms/step - loss: 0.6973 - mae: 0.6753 - val_loss: 0.6170 - val_mae: 0.6328

from tensorflow.keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)

```

```
history = model.fit(X_train, y_train, epochs=50, batch_size=32,
validation_split=0.2, callbacks=[early_stopping], verbose=1)
```

```
584/584 [=====] - 2s 4ms/step - loss: 0.6973 - mae: 0.6732 - val_loss: 0.6093 - val_mae: 0.6274
Epoch 2/50 [=====]
584/584 [=====] - 2s 4ms/step - loss: 0.6963 - mae: 0.6732 - val_loss: 0.6076 - val_mae: 0.6269
Epoch 3/50 [=====]
584/584 [=====] - 2s 4ms/step - loss: 0.6964 - mae: 0.6729 - val_loss: 0.6129 - val_mae: 0.6237
Epoch 4/50 [=====]
584/584 [=====] - 3s 4ms/step - loss: 0.6984 - mae: 0.6741 - val_loss: 0.6165 - val_mae: 0.6425
Epoch 5/50 [=====]
584/584 [=====] - 3s 5ms/step - loss: 0.6955 - mae: 0.6724 - val_loss: 0.6234 - val_mae: 0.6250
Epoch 6/50 [=====]
584/584 [=====] - 2s 4ms/step - loss: 0.6941 - mae: 0.6706 - val_loss: 0.6359 - val_mae: 0.6262
Epoch 7/50 [=====]
584/584 [=====] - 2s 4ms/step - loss: 0.6934 - mae: 0.6689 - val_loss: 0.5964 - val_mae: 0.6222
Epoch 8/50 [=====]
584/584 [=====] - 2s 4ms/step - loss: 0.6924 - mae: 0.6690 - val_loss: 0.6019 - val_mae: 0.6199
Epoch 9/50 [=====]
584/584 [=====] - 2s 4ms/step - loss: 0.6893 - mae: 0.6677 - val_loss: 0.6204 - val_mae: 0.6377
Epoch 10/50 [=====]
584/584 [=====] - 3s 5ms/step - loss: 0.6871 - mae: 0.6676 - val_loss: 0.6001 - val_mae: 0.6164
Epoch 11/50 [=====]
584/584 [=====] - 3s 5ms/step - loss: 0.6830 - mae: 0.6631 - val_loss: 0.6223 - val_mae: 0.6494
Epoch 12/50 [=====]
584/584 [=====] - 2s 4ms/step - loss: 0.6855 - mae: 0.6661 - val_loss: 0.6087 - val_mae: 0.6204
```

```
#function to test the model
# Generate a random index from the test set
random_index = np.random.choice(len(X_test))

# Predicted value
```

```

predicted_value =
model.predict(np.array([X_test.iloc[random_index]]))[0][0]
true_value = y_test.iloc[random_index]

# Create a colormap for the heatmap
cmap = plt.get_cmap('Blues')

# Plot the line heatmap
plt.figure(figsize=(8, 4))

# Add a heatmap background
heatmap = plt.imshow([np.arange(0, 7)], cmap=cmap,
aspect='auto', extent=[sizes[0] - 0.5, sizes[-1] + 0.5, 0, 1],
alpha=0.6)

# add the predicted value as a vertical line
plt.axvline(predicted_value, color='k', linestyle='--',
linewidth=1)

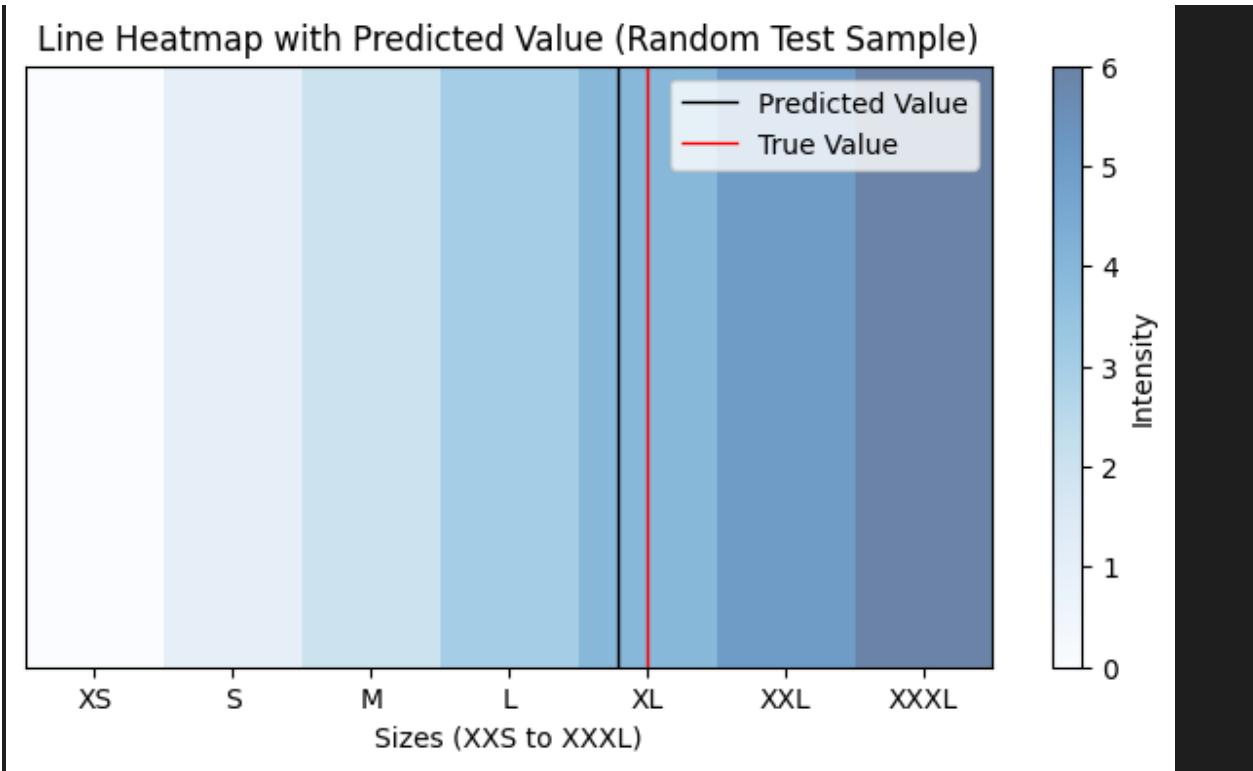
# add the true value as a vertical line
plt.axvline(true_value, color='r', linestyle='--', linewidth=1)

# Customize the plot
plt.xticks(sizes, sizes_labels)
plt.yticks([])
plt.xlabel('Sizes (XXS to XXXL)')
plt.title('Line Heatmap with Predicted Value (Random Test Sample)')
plt.legend(['Predicted Value', 'True Value'])

# Show the colorbar for the heatmap
cbar = plt.colorbar(heatmap)
cbar.set_label('Intensity')

# Display the plot
plt.show()

```



```
#testing the model with external data
# Function to calculate BMI
def calculate_bmi(weight, height):
    return weight / ((height / 100) ** 2)

# Function to normalize user input
def normalize_input(user_input, scaler):
    normalized_input = scaler.transform(user_input.reshape(1, -1))
    return normalized_input

# Function to predict size and provide recommendations using the TensorFlow model
def predict_size_and_recommend_tf(user_input, model, scaler, size_mapping):
    # Calculate BMI

    bmi = calculate_bmi(user_input[0], user_input[2])

    # Add BMI to user input
    user_input_with_bmi = np.append(user_input, bmi)
```

```

# Normalize input
normalized_input = normalize_input(user_input_with_bmi,
scaler)

# Predict size
predicted_size = model.predict(normalized_input)[0][0]

# Map predicted size to category
for size, encoded_size in size_mapping.items():
    if encoded_size == round(predicted_size):
        predicted_size_category = size
        break
    else:
        # Handle the case when predicted size is not in the
        categories
        # Find the two sizes between which the predicted size
falls
        sizes = list(size_mapping.keys())
        sizes.sort(key=lambda s: size_mapping[s])

        for i in range(len(sizes) - 1):
            if size_mapping[sizes[i]] < round(predicted_size) <
size_mapping[sizes[i+1]]:
                recommended_sizes = [sizes[i], sizes[i+1]]
                break
            else:
                # Edge case if the predicted size is greater than
the biggest size
                recommended_sizes = [sizes[5], sizes[6]]

        return f"The predicted size is not in the defined
categories. Recommended sizes: {', '.join(recommended_sizes)}"

    return f"Predicted size: {predicted_size_category}"

# User input
age = float(input("Enter age: "))
weight = float(input("Enter weight (kg): "))
height = float(input("Enter height (cm): "))

# Create user input array
user_input = np.array([weight, age, height])

```

```

# Predict size and provide recommendations using the TensorFlow
model
result_tf = predict_size_and_recommend_tf(user_input, model,
scaler, size_mapping)

# Print the result
print(result_tf)
Enter age: 21
Enter weight (kg): 58
Enter height (cm): 175
1/1 [=====] - 0s 22ms/step
Predicted size: M

#Saving model for deployment

#Convert the TensorFlow model to pickle format
import joblib
from google.colab import files

# Save the TensorFlow model to a pickle file
joblib.dump(model, 'tensorflow_model.pkl')

# Download the pickle file locally
files.download('tensorflow_model.pkl')

```

## 5.4.2 Flask endpoint for Size recommendation model:

The Flask application provides a size recommendation endpoint designed for both mobile and web applications. Users submit their weight, age, and height through a POST request to the `/submit` endpoint, where the input data is processed and validated. The application loads a pre-trained machine learning model and a scaler to normalize the input, then predicts the appropriate clothing size by calculating the user's BMI and mapping the prediction to a size category. The result is returned as a JSON

response, which can be easily consumed by mobile and web applications for seamless integration and user interaction.

## Code implementation:

```
from flask import Flask, render_template, request, jsonify
import pandas as pd
import pickle
import numpy as np

app = Flask(__name__)

# Define the columns for DataFrame
cols = ['weight', 'age', 'height']

# Define function to calculate BMI
def calculate_bmi(weight, height):
    return weight / ((height / 100) ** 2)

# Function to process form data
def get_data(weight, age, height):
    try:
        weight = float(weight)
        age = int(age)
        height = float(height)
    except ValueError:
        return None # Return None if conversion fails

    data = [[weight, age, height]] # Create a list of lists
    with one inner list containing the data
    d = pd.DataFrame(data, columns=cols) # Create DataFrame
    from the list of data and columns
    return d

# Function to load the trained model
def load_model(model_path):
    with open(model_path, 'rb') as file:
        model = pickle.load(file)
    return model

# Function to normalize user input
def normalize_input(user_input, scaler):
    normalized_input = scaler.transform(user_input)
    return normalized_input
```

```

# Function to predict size and provide recommendations
def predict_size_and_recommend(user_input, model, scaler,
size_mapping):
    # Calculate BMI for each row
    user_input['bmi'] = user_input.apply(lambda row:
calculate_bmi(row['weight'], row['height']), axis=1)

    # Normalize input
    normalized_input =
normalize_input(user_input.values.reshape(1, -1), scaler)

    # Predict size
    predicted_size = model.predict(normalized_input)[0]

    # Map predicted size to category
    for size, encoded_size in size_mapping.items():
        if encoded_size == predicted_size:
            predicted_size_category = size
            break
    else:
        # Handle the case when predicted size is not in the
        categories
        sizes = list(size_mapping.keys())
        sizes.sort(key=lambda s: size_mapping[s])

        for i in range(len(sizes) - 1):
            if size_mapping[sizes[i]] < predicted_size <
size_mapping[sizes[i+1]]:
                recommended_sizes = [sizes[i], sizes[i+1]]
                break
        else:
            recommended_sizes = [sizes[-2], sizes[-1]] # Use
the last two sizes as recommendations

    return {"predicted_size": f"{recommended_sizes[0]} or
{recommended_sizes[1]}"}}

    return {"predicted_size": predicted_size_category}

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/submit', methods=['POST'])
def submit():
    weight = request.form.get('weight')
    age = request.form.get('age')

```

```

height = request.form.get('height')
df = get_data(weight, age, height)

if df is not None:
    model = load_model("linear_regression_model (2).pkl")
    scaler = pickle.load(open("scaler (1).pkl", "rb"))
    size_mapping = {'XS': 1, 'S': 2, 'M': 3, 'L': 4, 'XL': 5, 'XXL': 6, 'XXXL': 7}

    prediction = predict_size_and_recommend(df, model,
scaler, size_mapping)
    # Return the prediction as JSON
    return jsonify(prediction)
else:
    return jsonify({"error": "Error processing form data.
Please check your input."})

if __name__ == "__main__":
    app.run(debug=True)

```

The screenshot shows the Postman interface with a POST request to `http://127.0.0.1:5000/submit`. The request body is set to `x-www-form-urlencoded` and contains three parameters: `weight`, `age`, and `height`, each with a value of 58, 21, and 175 respectively. The response is a 200 OK status with a JSON body containing the predicted size: `{"predicted_size": "S or M"}`.

### 5.4.3 Fashion recommendation model:

The provided code is for a fashion recommendation model using K-Nearest Neighbors (KNN) and Principal Component Analysis (PCA) for visually similar content filtering. The system loads and merges image data and product metadata, processes images using a pre-trained VGG16 model for feature extraction, and reduces the dimensionality of these features using PCA. The KNN model is trained to find visually similar products. The recommendation system takes an input image and returns IDs of similar fashion items. The model is saved as a pickle file for future use and deployed via a Flask application, ensuring it can recommend similar products accurately based on visual similarity and with external data to ensure robust recommendations.

#### Dataset:

The growing e-commerce industry presents a large dataset waiting to be scraped and researched upon. In addition to professionally shot high-resolution product images, there are multiple label attributes describing the product, which were manually entered while cataloging. To add to this, there is also descriptive text that comments on the product characteristics.

Each product is identified by an ID like 42431. A map to all the products can be found in styles.csv. From here, the image for this product can be fetched from images/42431.jpg and the complete metadata from styles/42431.json.

## Code Implementation:

```
import numpy as np
import pandas as pd
import os
import re
import tensorflow as tf
from threading import Thread
import time
from tqdm import tqdm
import matplotlib.pyplot as plt
import plotly.express as px
from plotly.offline import init_notebook_mode
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
load_img, img_to_array
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import Sequence
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
GlobalAveragePooling2D, Activation, Dropout, Flatten, Dense, Input,
Layer
from tensorflow.keras.applications import VGG16, ResNet50,
DenseNet201, Xception
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import plot_model
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau
from sklearn.metrics import accuracy_score

init_notebook_mode(connected=True)
```

## EDA and Visualization

```
images_df = pd.read_csv("../input/fashion-product-images-
dataset/fashion-dataset/images.csv")

styles_df = pd.read_csv("../input/fashion-product-images-
dataset/fashion-dataset/styles.csv", on_bad_lines='skip')

images_df['id'] = images_df['filename'].apply(lambda x:
x.replace(".jpg", "")).astype(int)
```

```
images_df
```

	filename	link	id
0	15970.jpg	http://assets.myntassets.com/v1/images/style/p...	15970
1	39386.jpg	http://assets.myntassets.com/v1/images/style/p...	39386
2	59263.jpg	http://assets.myntassets.com/v1/images/style/p...	59263
3	21379.jpg	http://assets.myntassets.com/v1/images/style/p...	21379
4	53759.jpg	http://assets.myntassets.com/v1/images/style/p...	53759
...	...	...	...
44441	17036.jpg	http://assets.myntassets.com/v1/images/style/p...	17036
44442	6461.jpg	http://assets.myntassets.com/v1/images/style/p...	6461
44443	18842.jpg	http://assets.myntassets.com/v1/images/style/p...	18842
44444	46694.jpg	http://assets.myntassets.com/v1/images/style/p...	46694
44445	51623.jpg	http://assets.myntassets.com/assets/images/516...	51623

44446 rows × 3 columns

```
data =
styles_df.merge(images_df, on='id', how='left').reset_index(drop=True)
data['filename'] = data['filename'].apply(lambda x:
os.path.join("../input/fashion-product-images-dataset/fashion-
dataset/images/",x))

image_files = os.listdir("../input/fashion-product-images-
dataset/fashion-dataset/images")

data['file_found'] = data['id'].apply(lambda x: f"{x}.jpg" in
image_files)

data = data[data['file_found']].reset_index(drop=True)

data.head()

   id    gender      masterCategory    subCategory      articleType
  baseColour  season       year  usage productDisplayName    filename
   link  file_found
```

```

0    15970 Men Apparel Topwear Shirts Navy Blue Fall
     2011.0 Casual Turtle Check Men Navy Blue Shirt
     ./input/fashion-product-images-dataset/fashio...
     http://assets.myntassets.com/v1/images/style/p...True
1    39386 Men Apparel Bottomwear Jeans Blue Summer 2012.0
     Casual Peter England Men Party Blue Jeans
     ./input/fashion-product-images-dataset/fashio...
     http://assets.myntassets.com/v1/images/style/p...True
2    59263 Women Accessories Watches Watches Silver
     Winter 2016.0 Casual Titan Women Silver Watch
     ./input/fashion-product-images-dataset/fashio...
     http://assets.myntassets.com/v1/images/style/p...True
3    21379 Men Apparel Bottomwear Track Pants Black Fall
     2011.0 Casual Manchester United Men Solid Black Track
     Pants ./input/fashion-product-images-dataset/fashio...
     http://assets.myntassets.com/v1/images/style/p...True
4    53759 Men Apparel Topwear Tshirts Grey Summer
     2012.0 Casual Puma Men Grey T-shirt ./input/fashion-
     product-images-dataset/fashio...
     http://assets.myntassets.com/v1/images/style/p...True

data.isnull().sum()

```

```

id      0
gender   0
masterCategory   0
subCategory   0
articleType   0
baseColour   15
season     21
year      1
usage     317
productDisplayName  7
filename   0
link       0
file_found  0
dtype: int64

```

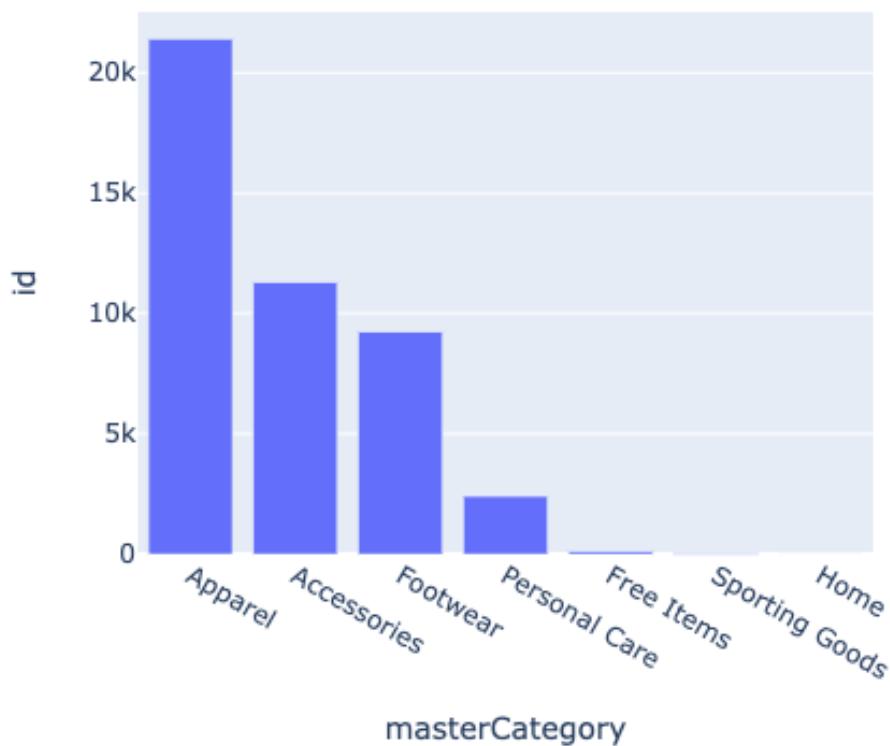
## Visualization

- Main Categories Count
- Sub Categories Count

- Products by Season Count
- Product Usage type Count

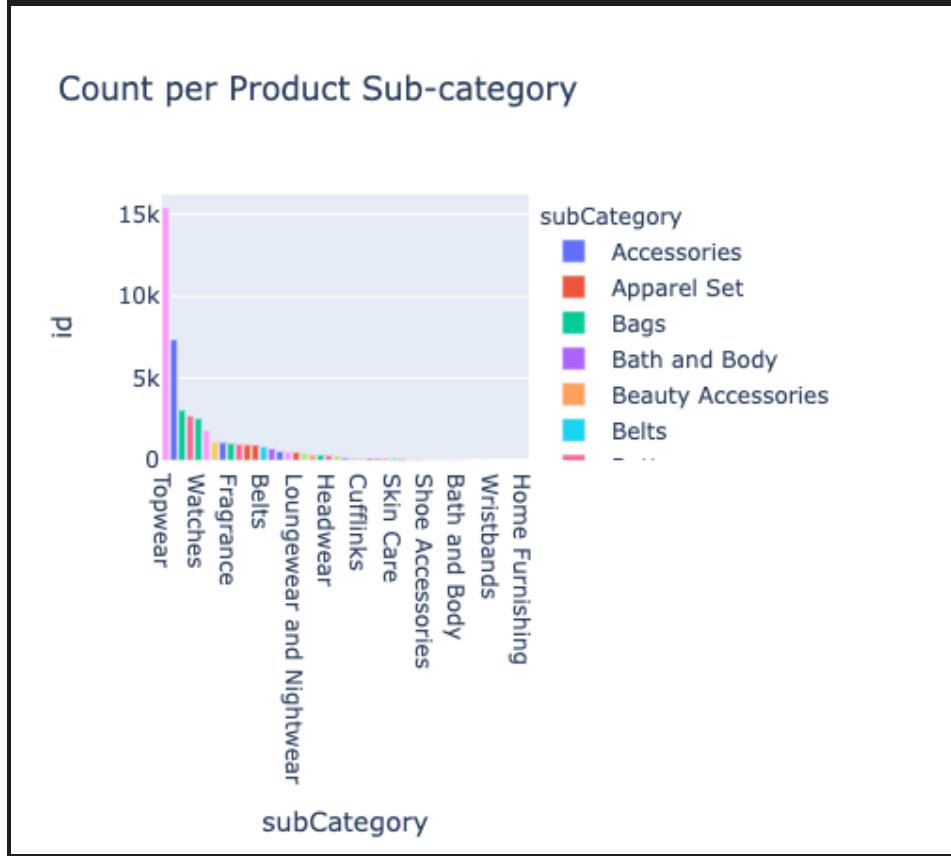
```
fig = px.bar(data.groupby('masterCategory').count().reset_index(),  
x='masterCategory',y='id',title='Count per Product Category')  
fig.update_layout(barmode='stack', xaxis={'categoryorder':'total  
descending'})
```

### Count per Product Category



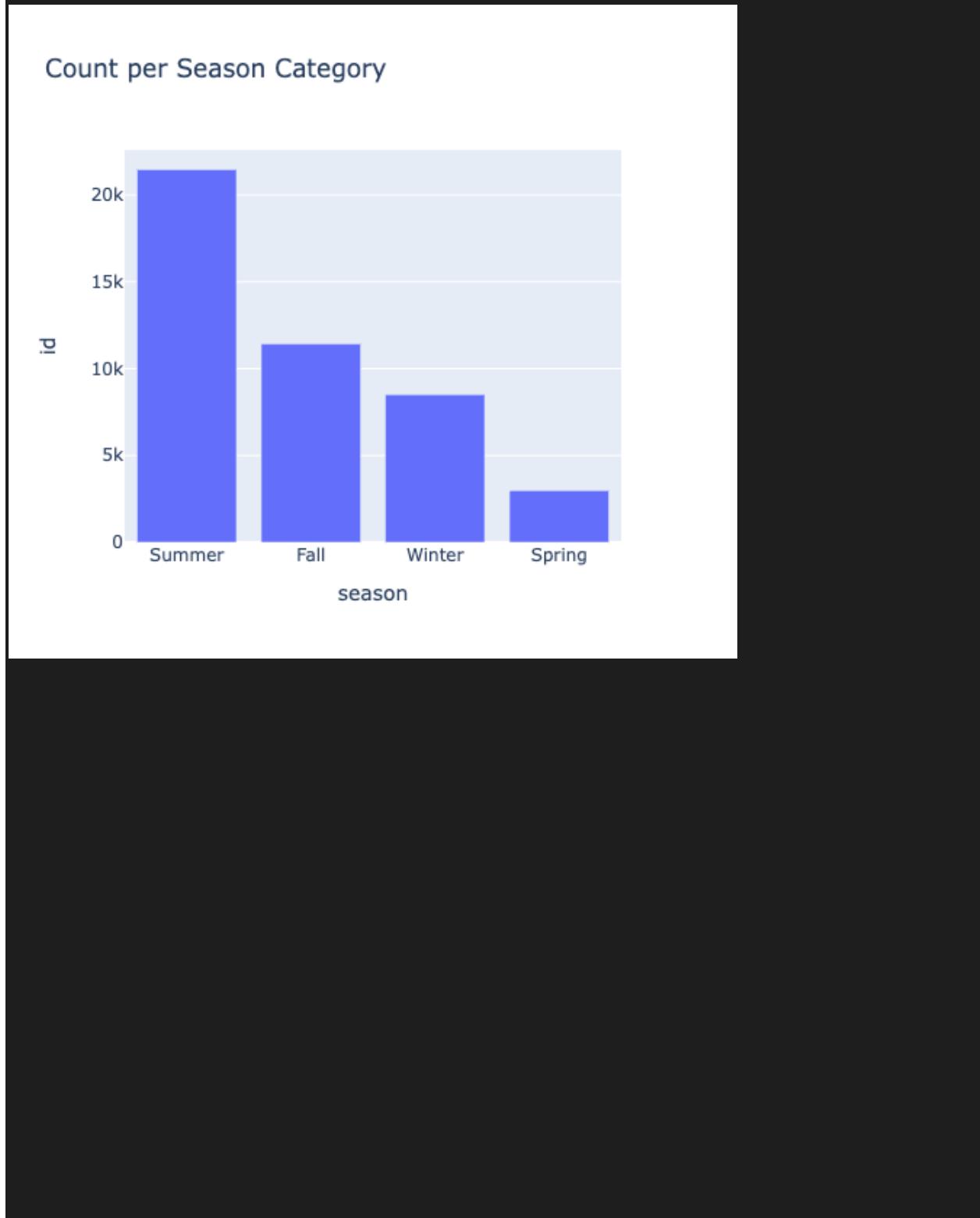
```
fig = px.bar(data.groupby('subCategory').count().reset_index(),
x='subCategory',y='id',title='Count per Product Sub-category',
color='subCategory')
```

```
fig.update_layout(barmode='stack', xaxis={'categoryorder':'total descending'})
```



```
fig = px.bar(data.groupby('season').count().reset_index(), x='season', y='id', title='Count per Season Category')
```

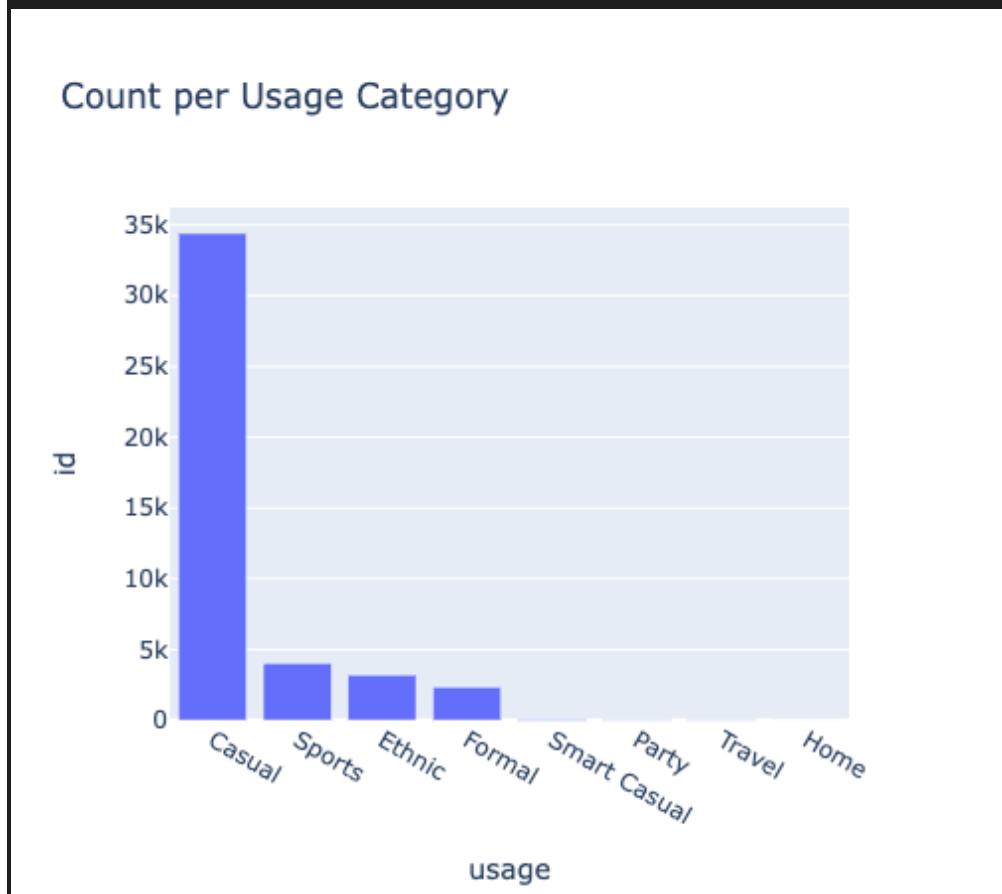
```
fig.update_layout(barmode='stack', xaxis={'categoryorder':'total  
descending'})
```



```

fig = px.bar(data.groupby('usage').count().reset_index(), x='usage',
y='id', title='Count per Usage Category')
fig.update_layout(barmode='stack', xaxis={'categoryorder':'total
descending'})

```



## Remove Unnecessary Columns

```

data.drop(columns=['productDisplayName','link','file_found'],inplace=True)
data

```

	id	gender	masterCategory	subCategory	articleType	baseColour	season	year	usage	filename	
0	15970		Men	Apparel	Topwear	Shirts	NAVY BLUE	Fall	2011.0	Casual	./input/fashion-product-images-dataset/fashio...

<b>id</b>	<b>gen der</b>	<b>masterC ategory</b>	<b>subCat egory</b>	<b>article Type</b>	<b>baseC olour</b>	<b>sea son</b>	<b>year</b>	<b>usa ge</b>	<b>filen ame</b>
1	393 86	Men	Apparel	Bottomwear	Jeans	Blue	Summer	201 2.0	Casual ..../input/fashion-product-images-dataset/fashio...
2	592 63	Women	Accessories	Watches	Watches	Silver	Winter	201 6.0	Casual ..../input/fashion-product-images-dataset/fashio...
3	213 79	Men	Apparel	Bottomwear	Track Pants	Black	Fall	201 1.0	Casual ..../input/fashion-product-images-dataset/fashio...
4	537 59	Men	Apparel	Topwear	Tshirts	Grey	Summer	201 2.0	Casual ..../input/fashion-product-images-dataset/fashio...
...	...	...	...	...	...	...	...	...	...
44 41 4	170 36	Men	Footwear	Shoes	Casual Shoes	White	Summer	201 3.0	Casual ..../input/fashion-product-images-dataset/fashio...
44 41 5	646 1	Men	Footwear	Flip Flops	Flip Flops	Red	Summer	201 1.0	Casual ..../input/fashion-product-images-dataset/fashio...

<b>id</b>	<b>gen der</b>	<b>masterC ategory</b>	<b>subCat egory</b>	<b>article Type</b>	<b>baseC olour</b>	<b>sea son</b>	<b>year</b>	<b>usa ge</b>	<b>filen ame</b>
44	188	Men	Apparel	Topwe ar	Tshirts	Blu e	Fall	201 1.0	Casu al
41	42								..../input/f ashion- product- images- dataset/f ashio...
6									
44	466	Women	Person al Care	Fragra nce	Perfu me and Body Mist	Blu e	Spri ng	201 7.0	Casu al
41	94								..../input/f ashion- product- images- dataset/f ashio...
7									
44	516	Women	Access ories	Watch es	Watch es	Pin k	Wint er	201 6.0	Casu al
41	23								..../input/f ashion- product- images- dataset/f ashio...
8									

44419 rows × 10 columns

## Train-Val Split

```
data = data.sample(frac=1).reset_index(drop=True)
n = len(data)
train = data.iloc[:int(n*0.8),:]
val = data.iloc[int(n*0.8):,:].reset_index(drop=True)
```

## Data Generator

```
datagen = ImageDataGenerator(rescale=1/255.)

train_generator = datagen.flow_from_dataframe(dataframe=train,
                                              target_size=(256, 256),
                                              x_col='filename',
                                              class_mode=None,
                                              batch_size=32,
                                              shuffle=False,
                                              classes=['images'])

val_generator = datagen.flow_from_dataframe(dataframe=val,
```

```

        target_size=(256, 256),
        x_col='filename',
        class_mode=None,
        batch_size=32,
        shuffle=False,
        classes=['images'])

Found 35535 validated image filenames.

```

```
Found 8884 validated image filenames.
```

## Feature Extraction: Pre-trained VGG16

```
base_model = VGG16(include_top=False, input_shape=(256, 256, 3))
```

```

model = Sequential()
for layer in base_model.layers:
    model.add(layer)
model.add(GlobalAveragePooling2D())
model.summary()

```

Downloading data from [https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16\\_weights\\_tf\\_dim\\_ordering\\_tf\\_kernels\\_notop.h5](https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5)

58889256/58889256 ━━━━━━━━━━━━ 1s 0us/step

Model: "sequential"

Layer (type)	Output Shape	Param #
block1_conv1 (Conv2D)	(None, 256, 256, 64)	1,792
block1_conv2 (Conv2D)	(None, 256, 256, 64)	36,928
block1_pool (MaxPooling2D)	(None, 128, 128, 64)	0
block2_conv1 (Conv2D)	(None, 128, 128, 128)	73,856
block2_conv2 (Conv2D)	(None, 128, 128, 128)	147,584

block2_pool (MaxPooling2D)	(None, 64, 64, 128)	0
block3_conv1 (Conv2D)	(None, 64, 64, 256)	295,168
block3_conv2 (Conv2D)	(None, 64, 64, 256)	590,080
block3_conv3 (Conv2D)	(None, 64, 64, 256)	590,080
block3_pool (MaxPooling2D)	(None, 32, 32, 256)	0
block4_conv1 (Conv2D)	(None, 32, 32, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 32, 32, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 32, 32, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 16, 16, 512)	0
block5_conv1 (Conv2D)	(None, 16, 16, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 16, 16, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 16, 16, 512)	2,359,808

```

| block5_pool (MaxPooling2D) | (None, 8, 8, 512) | 0 |
|-----|
| global_average_pooling2d | (None, 512) | 0 |
| (GlobalAveragePooling2D) | | |
|-----|
Total params: 14,714,688 (56.13 MB)
Trainable params: 14,714,688 (56.13 MB)
Non-trainable params: 0 (0.00 B)

```

## Extracting Features of Training and Validation Set

```

train_features = model.predict(train_generator, verbose=1)
val_features = model.predict(val_generator, verbose=1)

```

## Dimensionality Reduction

```
from sklearn.decomposition import PCA
```

### Illustration of how PCA finds the axes where the within data variability will be maximum

```

pca = PCA(2)
pca.fit(train_features)
train_pca = pca.transform(train_features)
test_pca = pca.fit_transform(val_features)
train_pca = pd.DataFrame(train_pca)
train = train.iloc[:,0:10]
train = train.merge(train_pca, how='left', left_index=True,
right_index=True)

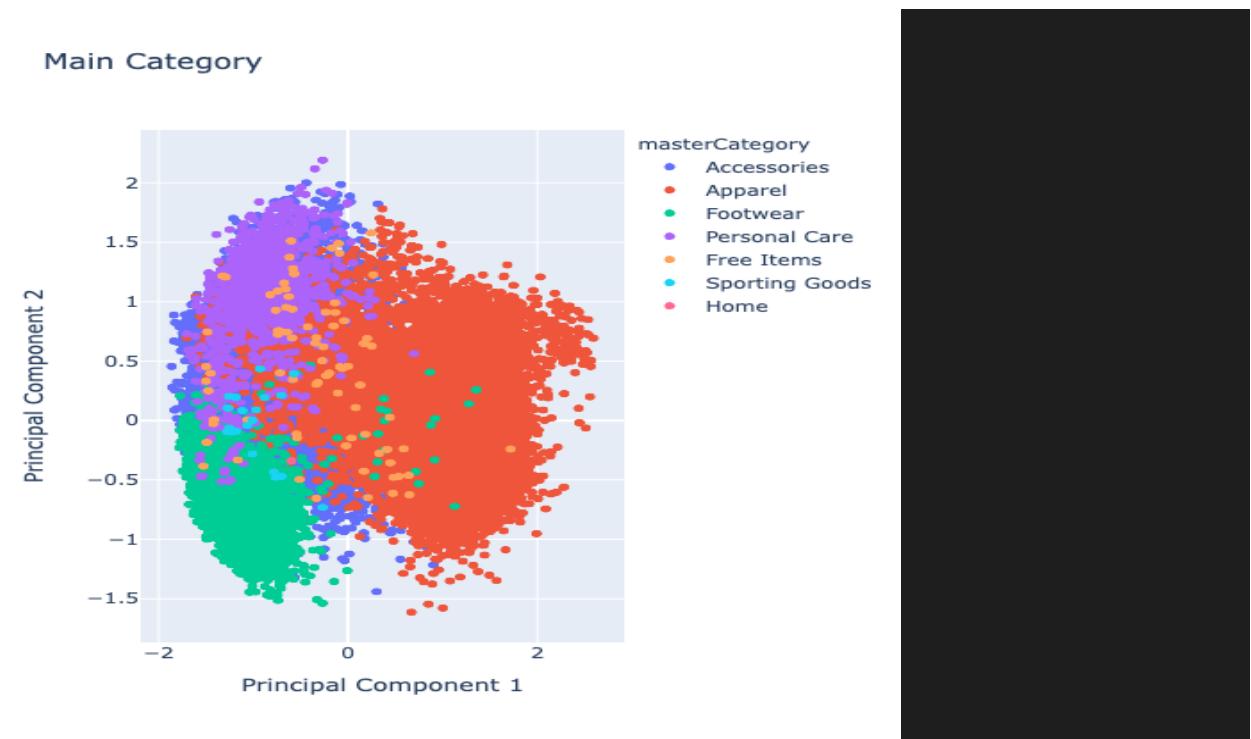
```

## Visualization: Principal Components

```

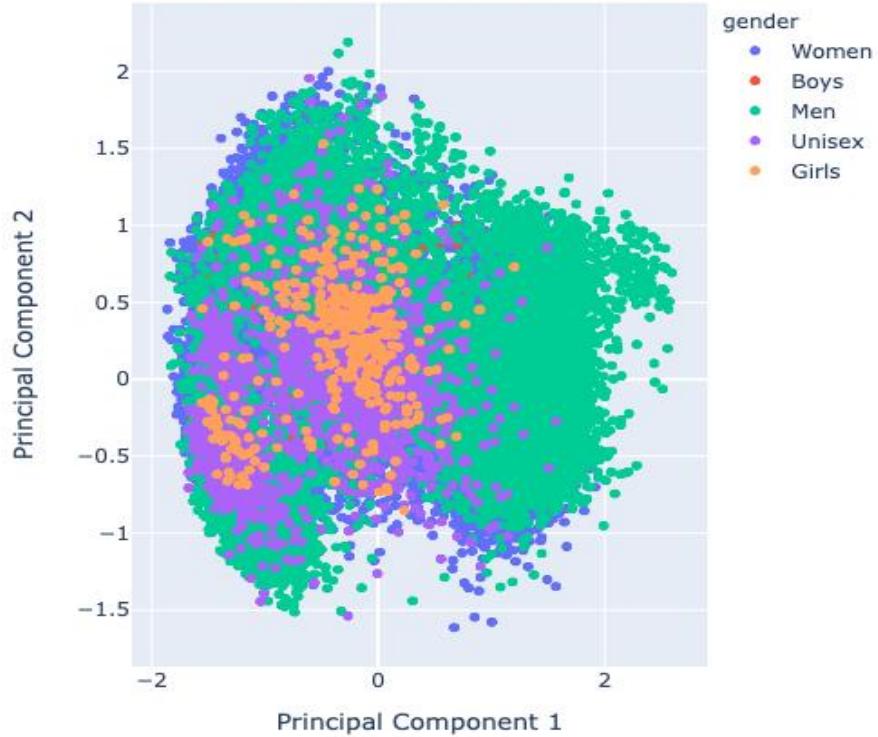
fig = px.scatter(train, x=0, y=1, color="masterCategory", title='Main
Category', height=600, labels={
    "0": "Principal Component 1",
    "1": "Principal Component 2"})
fig.show()

```



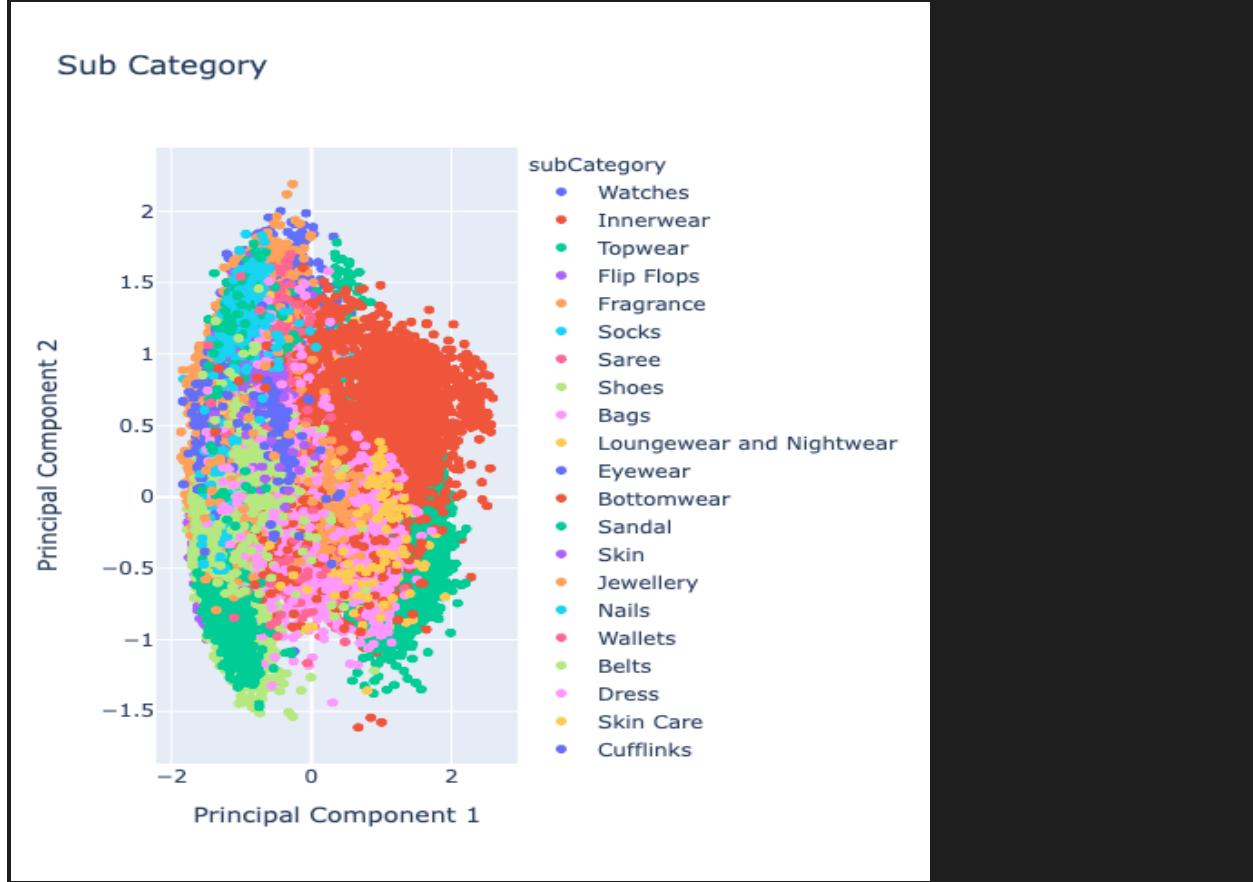
```
fig = px.scatter(train, x=0, y=1, color="gender", title='Gender',
height=600, labels={
    "0": "Principal Component 1",
    "1": "Principal Component 2"})
fig.show()
```

Gender



```
fig = px.scatter(train, x=0, y=1, color="subCategory", title='Sub Category', height=600, labels={  
    "0": "Principal Component 1",  
    "1": "Principal Component 2"})
```

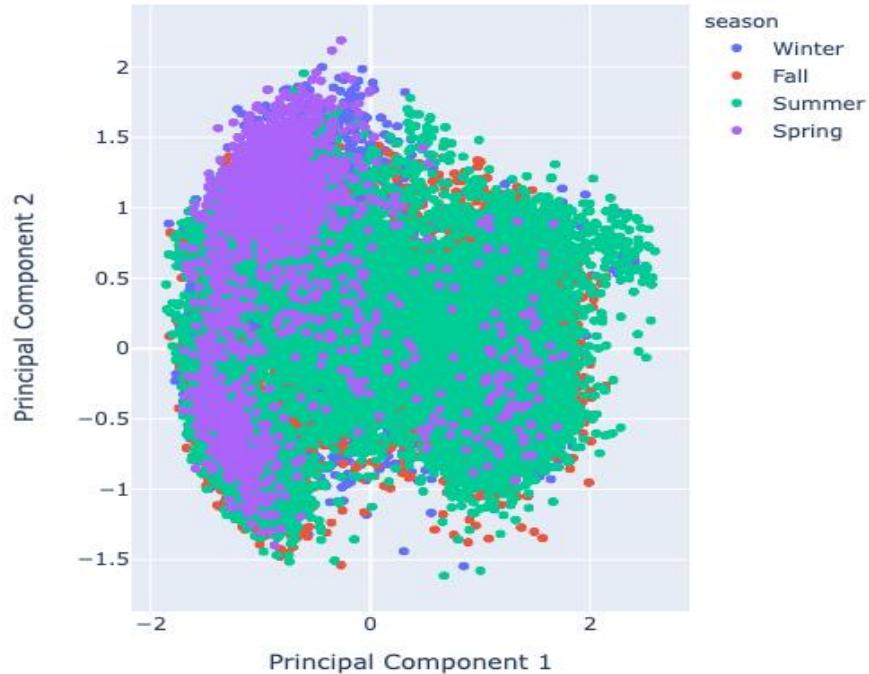
```
fig.show()
```



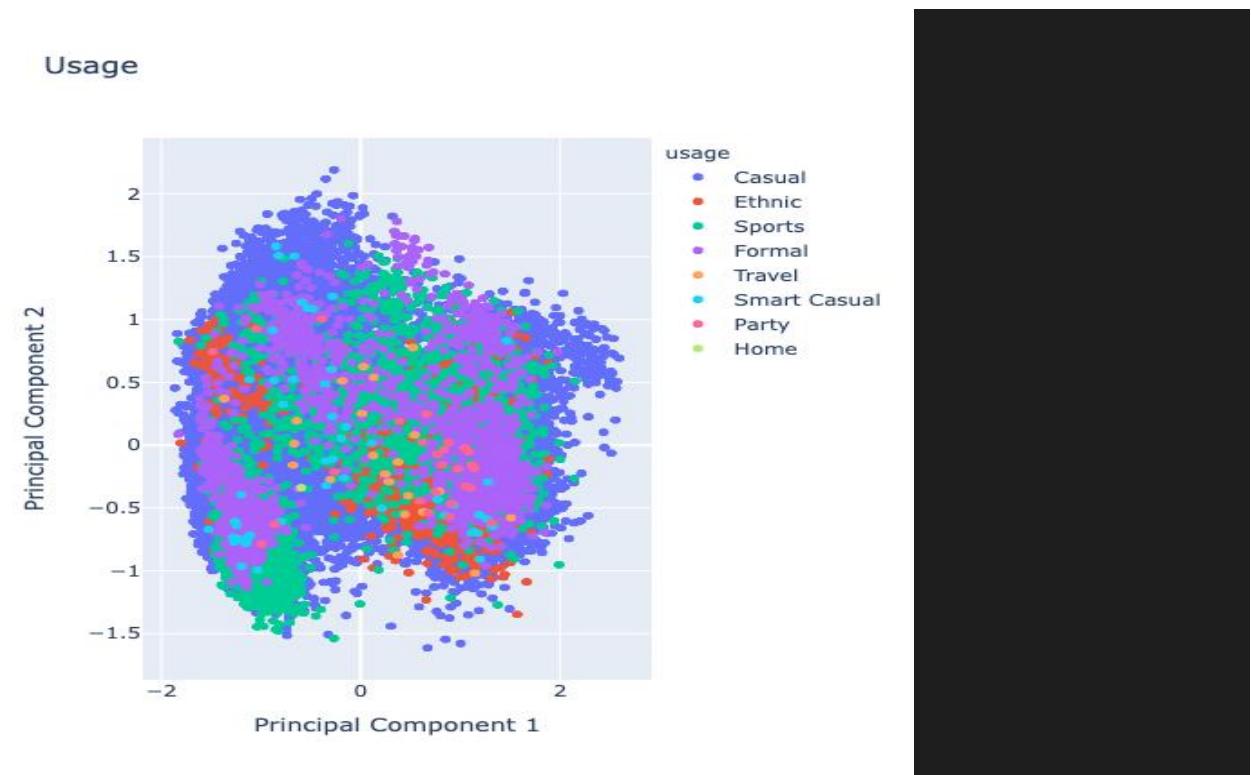
```
fig = px.scatter(train[train['season'].notna()], x=0, y=1,  
color="season", title='Season', height=600, labels={  
    "0": "Principal Component 1",  
    "1": "Principal Component 2"})
```

```
fig.show()
```

Season



```
fig = px.scatter(train[train['usage'].notna()], x=0, y=1,
color="usage", title='Usage', height=600, labels={
    "0": "Principal Component 1",
    "1": "Principal Component 2"})
fig.show()
```

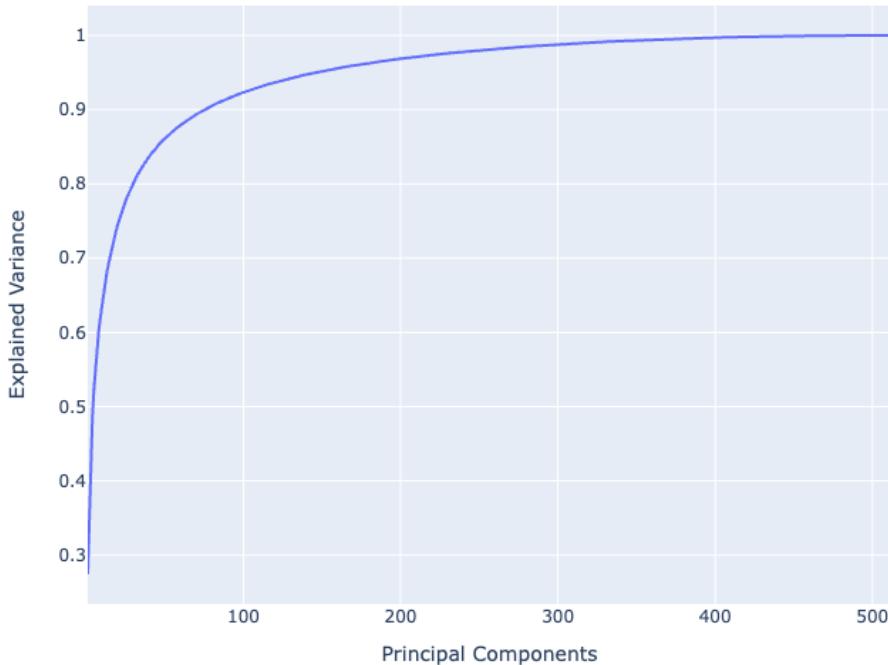


```
pca = PCA()
pca.fit(train_features)
train_pca = pca.transform(train_features)
variance_explained = np.cumsum(pca.explained_variance_ratio_)
pcs = range(1,len(variance_explained)+1)
```

## Dimensionality Reduction: 512 -> 313

```
px.line(x = pcs, y = variance_explained, title = 'Principal Components Cumulative Explained Variance', height=600, labels={
    "x": "Principal Components",
    "y": "Explained Variance"})
```

Principal Components Cumulative Explained Variance



```
val_pca = pca.fit_transform(val_features)[:, :313]
val_pca = pd.DataFrame(val_pca)
val = val.iloc[:, 0:10]
val = val.merge(val_pca, how='left', left_index=True,
right_index=True)

X = val.iloc[:, -313:]
y = val['id']
```

## K-Nearest Neighbours

```
from sklearn.neighbors import KNeighborsClassifier

neigh = KNeighborsClassifier(n_neighbors=6)
neigh.fit(X, y)
```

```

def read_img(image_path):
    image = load_img(image_path, target_size=(256, 256, 3))
    image = img_to_array(image)
    image = image/255.
    return image

# Accuracy
train_preds = neigh.predict(X_train)
val_preds = neigh.predict(X_val)
train_acc = accuracy_score(y_train, train_preds)
val_acc = accuracy_score(y_val, val_preds)
print(f"Train Accuracy: {train_acc}")
print(f"Validation Accuracy: {val_acc}")

```

## Results

```

# Displaying Results and Similar Image IDs
# Visualize Similar Products
for _ in range(10):
    i = random.randint(0, len(val) - 1)
    img1 = read_img(val.loc[i, 'filename'])
    dist, index = neigh.kneighbors(X=X.iloc[i, :].values.reshape(1, -
1))
    similar_ids = val.loc[index[0], 'id'].tolist()
    print(f"Input Image ID: {val.loc[i, 'id']}")
    print(f"Similar Image IDs: {similar_ids}")

    plt.figure(figsize=(4, 4))
    plt.imshow(img1)
    plt.title("Input Image")

    plt.figure(figsize=(20, 20))
    for j in range(1, 6):
        plt.subplot(1, 5, j)
        plt.subplots_adjust(hspace=0.5, wspace=0.3)
        image = read_img(val.loc[index[0][j], 'filename'])
        plt.imshow(image)
        plt.title(f'Similar Product #{j}')
    plt.show()

```

Input Image ID: 18851

Similar Image IDs: [18851, 18849, 18843, 18224, 31279, 34317]

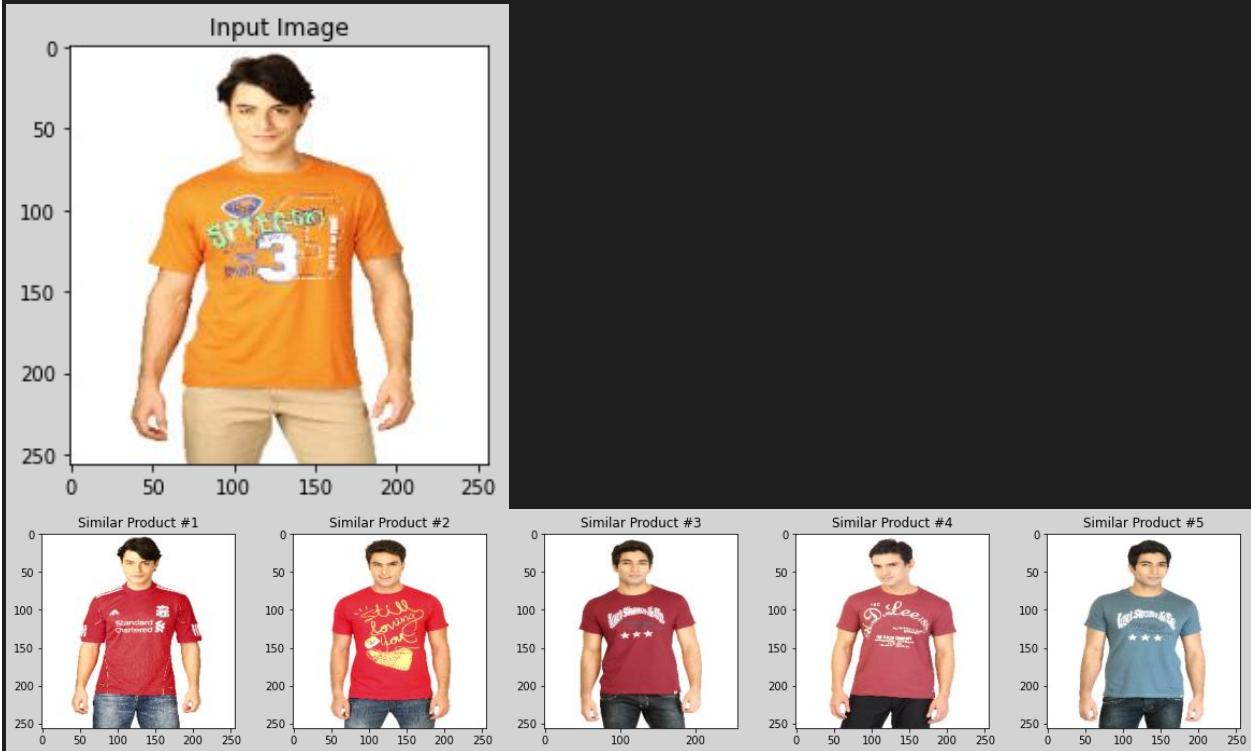


Input Image ID: 23222

Similar Image IDs: [23222, 23218, 10431, 10433, 18749, 23310]

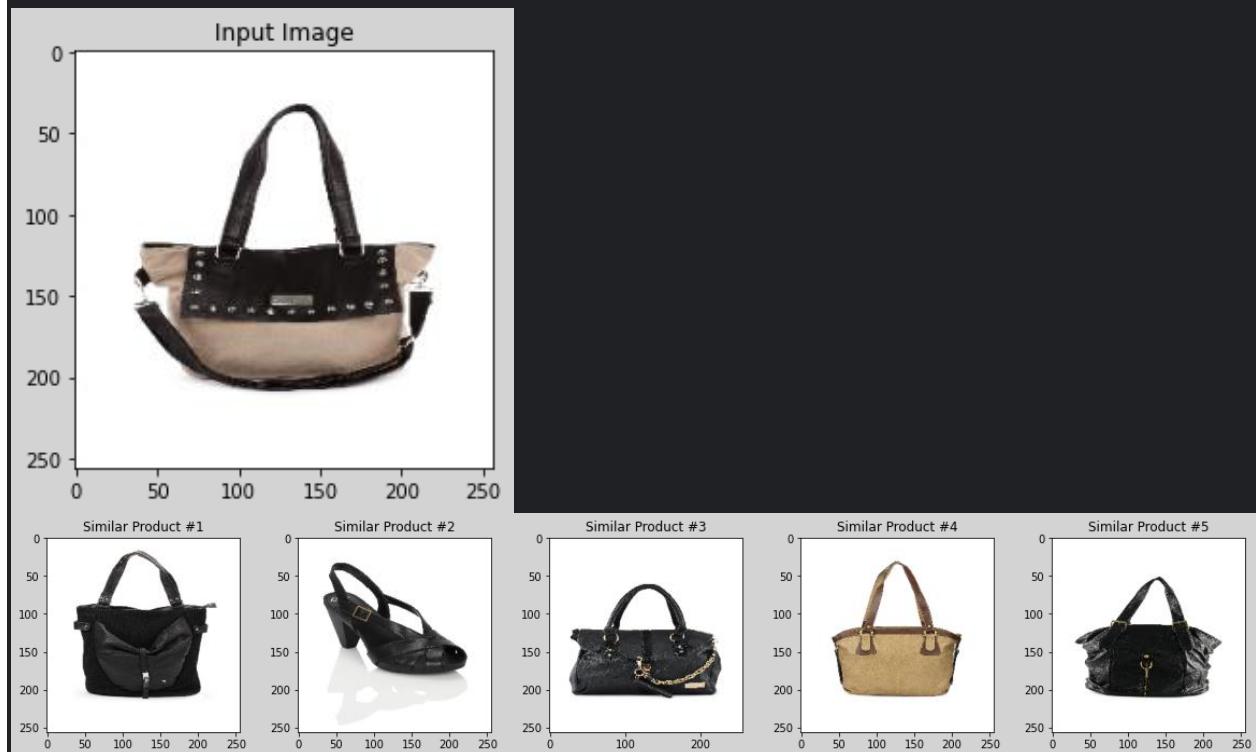


Input Image ID: 8556  
Similar Image IDs: [8556, 8410, 12631, 16439, 11771, 16430]



Input Image ID: 25192

Similar Image IDs: [25192, 46973, 34071, 17121, 37406, 21522]



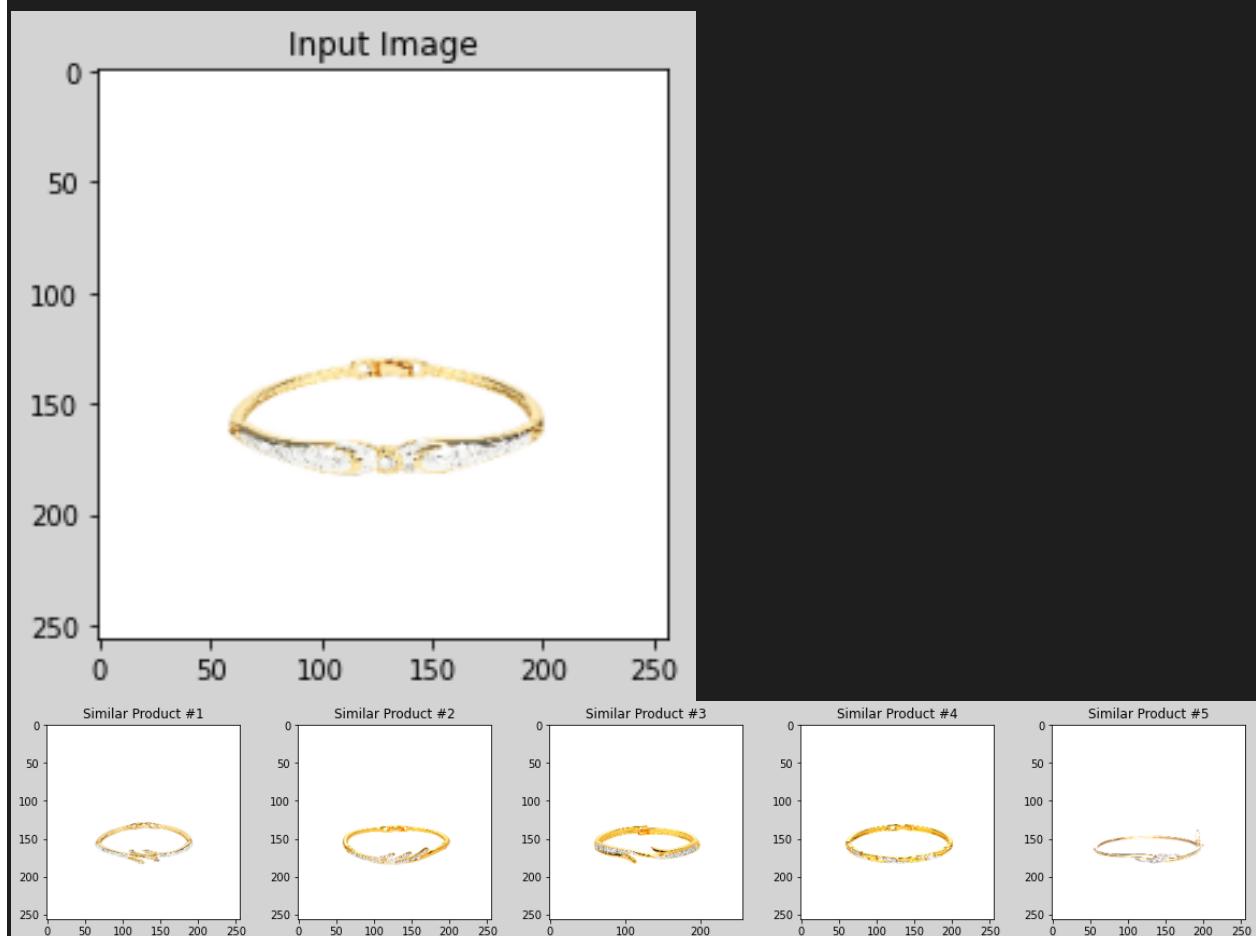
Input Image ID: 5384

Similar Image IDs: [5384, 15364, 15512, 47179, 13133, 38783]



Input Image ID: 59784

Similar Image IDs: [59784, 59779, 42637, 42632, 42639, 48747]



## Saving the model for deployment

```
# Save the PCA components and KNN model
with open('final_pca_components.pkl', 'wb') as pca_file:
    pickle.dump(pca, pca_file)

with open('final_knn_model.pkl', 'wb') as knn_file:
    pickle.dump(neigh, knn_file)
```

## 5.4.4 Flask endpoint for fashion recommendation model:

The provided code is endpoint for a Flask web application that provides fashion recommendations based on a given image URL. It uses a VGG16 model for feature extraction, PCA for dimensionality reduction, and a KNN model to find similar fashion items. When a POST request with an image URL is made to the `/similar\_images` endpoint, the app processes the image, extracts features, and returns IDs of recommended fashion items, ensuring robust input validation and handling errors appropriately. The models and PCA components are pre-loaded from pickle files, and external data ensures accurate image processing.

### Code Implementation:

```
from flask import Flask, request, jsonify
from tensorflow.keras.preprocessing.image import load_img,
img_to_array
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalAveragePooling2D
import numpy as np
import pickle
import requests
from io import BytesIO

app = Flask(__name__)

# Load PCA Components and KNN Model
with open('final_pca_components2.pkl', 'rb') as pca_file:
    pca = pickle.load(pca_file)

with open('final_knn_model2.pkl', 'rb') as knn_file:
    knn_model = pickle.load(knn_file)

# Load the pre-trained model
base_model = VGG16(include_top=False, input_shape=(256, 256, 3))
model = Sequential()
for layer in base_model.layers:
```

```

        model.add(layer)
model.add(GlobalAveragePooling2D())
model.summary()

# Function to Read and Process Image
def read_img_from_url(image_url):
    response = requests.get(image_url)
    image = load_img(BytesIO(response.content),
target_size=(256, 256))
    image = img_to_array(image)
    image = image / 255.
    return image

@app.route('/similar_images', methods=['POST'])
def get_similar_images():
    if not request.is_json:
        return jsonify({'error': 'Invalid input'}), 400

    data = request.get_json()
    if 'img' not in data:
        return jsonify({'error': 'No image URL provided'}), 400

    image_url = data['img']
    try:
        image = read_img_from_url(image_url)
    except Exception as e:
        return jsonify({'error': f'Failed to process image: {str(e)}'}), 400

    # Extract features from the image
    features = model.predict(np.array([image]))
    pca_features = pca.transform(features)

    # Find similar images using KNN
    dist, indices = knn_model.kneighbors(pca_features.reshape(1,
-1))
    similar_ids = indices[0].tolist()

    return jsonify({'similar_ids': similar_ids})

if __name__ == '__main__':
    app.run(debug=True)

```

# Postman Request:

The screenshot shows the Postman application interface. At the top, there's a navigation bar with links for Home, Workspaces, API Network, and a search bar. On the left, there's a sidebar with sections for Collections, Environments, and History. The main area shows a POST request to `http://127.0.0.1:5000/similar_images`. The 'Params' tab is selected, showing a single query parameter named 'Key'. Below the request details, the response section displays the following JSON data:

```
1 {  
2     "similar_ids": [  
3         28638,  
4         14188,  
5         13054,  
6         34079,  
7         29793,  
8         18925  
9     ]  
10 }
```

The response status is 200 OK, with a time of 1555 ms and a size of 219 B. There are tabs for Body, Cookies, Headers (5), and Test Results, with 'Pretty' selected. At the bottom, there are various navigation and utility buttons.

382

# Chapter 6

## **6.1 Conclusion:**

In conclusion, the "On Budget" project offers a transformative approach to modern shopping by addressing critical challenges faced by users in today's market. By seamlessly integrating online and offline stores, the platform enables users to browse and purchase effortlessly from any location. The implementation of AI-driven features, such as personalized recommendations, stable diffusion for custom t-shirt designs, and image recognition technology, significantly enhances the shopping experience, catering to individual preferences and needs. Additionally, transparent shipping policies and AI-optimized shipping options ensure cost-effective and efficient deliveries, further reducing user frustration. With a comprehensive suite of tools and technologies, including an AI-powered chatbot for immediate assistance, the "On Budget" project aims to revolutionize the shopping landscape, fostering higher user satisfaction and engagement. Ultimately, this project empowers users to shop smarter, within their budget, and with greater convenience, paving the way for a more intuitive and enjoyable shopping experience.

## **6.2 Future work:**

### **Shipping Policy and Options:**

Recommending various shipping companies along with their prices gives users the flexibility to choose the shipping option that best fits their needs and budget. There will be three shipping categories: basic, express, and luxury. Transparent shipping policies enhance trust and allow users to have more control over their shopping experience.

# Chapter 7

# References:

- <https://flutter.dev/>
- <https://docs.flutter.dev/development/ui/advanced/splash-screen>
- <https://docs.flutter.dev/development/data-and-backend/firebase>
- <https://firebase.flutter.dev/docs/overview/>
- <https://firebase.flutter.dev/docs/storage/overview>
- <https://firebase.flutter.dev/docs/firestore/overview>
- <https://dart.dev/guides>
- [https://www.youtube.com/Firebase/](https://www.youtube.com/Firebase)
- <https://firebase.google.com/podcasts/firebase-show>
- <https://getbootstrap.com/>
- <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
- <https://scikit-learn.org/stable/index.html>
- <https://huggingface.co/docs>
- <https://cloud.google.com/colab/docs/>
- <https://www.kaggle.com/docs/api>
- <https://www.tensorflow.org/>
- <https://www.tensorflow.org/guide>