CustomFit

Personalized Clothing E-Commerce Mobile App

(Industrial Project)

Undertaken By:

Abdul Wassay

REG. NO. CIIT/SP21-BCS-031/WAH

Syed Ali Asad

REG. NO. CIIT/SP21-BCS-036/WAH

Supervised By:

Dr. Samia Riaz

DEPARTMENT OF COMPUTER SCIENCES

COMSATS UNIVERSITY ISLAMABAD, WAH CAMPUS

WAH CANTT – PAKISTAN

SESSION 2021-2025

Table of Contents

1 Introduction **1**
1.1 System Introduction 1
1.2 Background of the System 1
1.3 Objectives of the System 1
1.4 Significance of the System 1

2 Overall Description **3**
2.1 Product Perspective 3
2.2 Product Scope 3
2.3 Product Functionality 3
2.4 Users and Characteristics 4
2.4.1 Administrator 4
2.4.2 Customers 5
2.4.3 Tailors 5
2.5 Operating Environment 5

3. Specific Requirments **7**
3.1 Functional Requirements 6
3.1.1 Clothing Design and Customization: 6
3.1.2 Measurement Management: 6
3.1.3 Order Tracking and Management: 6
3.1.4 Secure Payment Processing: 6
3.1.5 Order History and Favourites: 6
3.1.6 Feedback and Reviews: 6
3.2 Behavior Requirements 7
3.3 External Interface Requirements 8
3.3.1 User Interfaces 8
3.3.2 Hardware Interfaces 14
3.3.3 Software Interfaces 14

4. Other Non-Functional Requirements **19**
4.1 Performance Requirements 15
4.1.1 Response Time 15
4.1.2 Availability 15
4.1.3 Accuracy 15
4.2 Safety and Security Requirements 15
4.3 Software Quality Attributes 15
4.3.1 Usability 15
4.3.2 Reliability 16
4.3.3 Portability 16

5. Design Discription **21**
5.1 Composite Viewpoint 17
5.2 Logical Viewpoint 18
5.3 Information Viewpoint 19
5.4 Interaction Viewpoint 20
5.4.1 Customer Sequence Diagram 20
5.4.2 Tailor Sequence Diagram 21

5.4.3 Administrator Diagram 22
5.5 State Dynamics Viewpoint 23
5.5.1 State Dynamic of the Customer 23
5.5.2 State Dynamic of Tailor 24
5.5.3 State Dynamic of Administrator 25
5.6 Algorithm Viewpoint 26
5.6.1 Algorithmic Viewpoint of Login 26
5.6.2 Algorithmic Viewpoint of Customer 27
5.6.3 Algorithmic Viewpoint of Tailor 34
5.6.4 Algorithmic Viewpoint of Administrator 39

List of Figures

Figure 3.1- Use Case Diagram of CustomFit........ 7
Figure 3.2 - CustomFit Login Screen........ 9
Figure 3.3 - Dashboard Screen........ 11
Figure 3.4 - Product Detail Screen........ 12
Figure 3.5 - Cart Screen........ 13
Figure 5.1 - Activity Diagram CustomFit........ 17
Figure 5.2 - Class Diagram CustomFit........ 18
Figure 5.3 - ER Diagram CustomFit........ 19
Figure 5.4 - Sequence Diagram of Customer........ 20
Figure 5.5 - Sequence Diagram Tailor........ 21
Figure 5.6 - Sequence Diagram administrator........ 22
Figure 5.7 - State Dynamic of Customer........ 23
Figure 5.8 - State Dynamic of Tailor........ 24
Figure 5.9 - State Dynamic of Administrator........ 25

List of Tables

Table 3.1- Software Interfaces 14
Table 5.1-Algorithmic Viewpoint of Login........ 26
Table 5.2– Algorithmic View Point of Customer 27
Table 5.3 – Algorithmic Viewpoint of Tailor........ 34
Table 5.4- Algorithmic Viewpoint of Administrator........ 39

1 Introduction

1.1 System Introduction

The "Custom Fit" software aims to revolutionize the way customers interact with and purchase customizable products, particularly in the fashion industry. It provides a platform for users to order customized products tailored to their specific preferences and measurements. The software streamlines the ordering process, integrates secure payment methods, utilizes cutting-edge computer vision and machine learning techniques for accurate body measurements, and offers a user-friendly product catalog interface.

1.2 Background of the System

There are existing software solutions in the market catering to customization and online shopping experiences. However, "Custom Fit" stands out due to its unique combination of features, including precise body measurements through computer vision and ML algorithms, seamless integration with React Native for a cross-platform mobile experience, and a comprehensive product catalog. Unlike traditional off-the-shelf products, "Custom Fit" provides a personalized and intuitive shopping journey, enhancing customer satisfaction and loyalty.

1.3 Objectives of the System

The main objective is to create a CustomFit project is to provide more effective environment and precise measurements.

1. Simplify the process of ordering customized products for users.
2. Ensure accurate body measurements through computer vision and ML algorithms.
3. Integrate secure payment methods for seamless transactions.
4. Provide an intuitive and visually appealing product catalog interface.

1.4 Significance of the System

The system CustomFit can offer many benefits to e-commerce industry such as,

- **Tailored Fit:** CustomFit offers personalized clothing solutions, ensuring each garment fits perfectly to the individual's measurements. This eliminates the need for standard sizing and enhances comfort and confidence for the wearer.
- **Time Efficiency:** CustomFit automates the measurement process, reducing the time required for manual measurements and fittings. This saves both customers and tailors

valuable time, enabling faster turnaround for clothing orders.

- **Precision:** By utilizing computer vision and machine learning algorithms, CustomFit ensures precise measurements, resulting in accurate and consistent fitting clothing. This minimizes the risk of errors and ensures customer satisfaction with their customized garments.
- **Enhanced Customer Experience:** With CustomFit, customers enjoy a seamless and convenient shopping experience. They can easily input their measurements through the app, browse customized options, and receive perfectly fitting garments, leading to increased satisfaction and loyalty.
- **Cost Savings:** CustomFit reduces the need for multiple fittings and alterations, saving customers money on tailoring expenses. Additionally, businesses can optimize inventory management and production processes, leading to cost savings in the long run.
- **Innovation:** CustomFit represents a cutting-edge solution in the fashion industry, leveraging technology to revolutionize the way clothing is tailored and purchased. By embracing innovation, businesses can stay ahead of the curve and attract tech-savvy customers.

2 Overall Description

2.1 Product Perspective

"Custom Fit" is a standalone software product designed to revolutionize the customization and online shopping experience. It interacts with users through a user-friendly interface and connects with external systems for payment processing and order fulfillment.

2.2 Product Scope

The product scope of CustomFit is features and functions related to online shopping and precise measurement. Some of the specific functionalities include,

- **Customized Fitting Experience:** CustomFit provides a platform for users to create personalized clothing orders tailored to their unique measurements and preferences.
- **Comprehensive Measurement Management:** CustomFit enables users to manage their measurement profiles securely within the platform. Users can input, update, and store their measurements, ensuring accuracy and consistency for future orders.
- **Integration Capabilities:** CustomFit is designed to seamlessly integrate with existing e-commerce platforms, payment gateways, and inventory management systems. This ensures smooth operation and data synchronization across various business processes.
- **Analytics and Insights:** CustomFit offers analytical tools to extract valuable insights from user data and order patterns. This feature enables businesses to identify trends, preferences, and areas for improvement, ultimately optimizing product offerings and customer experiences.

2.3 Product Functionality

Product functionality will define every function of our app:

- **Clothing Design and Customization:**

  CustomFit enables users to design and customize their clothing items, including selecting fabrics, colors, styles, and additional features such as pockets or buttons.

- **Management Measurements:**

  CustomFit provides tools for users to input, store, and manage their body measurements within the app. This ensures accurate and personalized fitting for each clothing order.

- **Order Tracking and Management:**

  Users can track the status of their orders in real-time, from placement to delivery. CustomFit also allows users to manage order details, make modifications, and communicate with tailors throughout the production process.

- **Secure Payment Processing:**

  CustomFit integrates secure payment gateways, enabling users to make payments for their orders within the app. Various payment methods are supported, including credit/debit cards, digital wallets, and bank transfers.

- **Order History and Favorites:**

  Users can view their order history, including past purchases and measurements, within the app. CustomFit also allows users to save favorite designs or clothing configurations for future reference.

- **Feedback and Reviews:**

  CustomFit allows users to provide feedback and reviews for their orders and overall experience with the app. This helps improve service quality and provides valuable insights for future enhancements.

2.4 Users and Characteristics

In CustomFit, there are two types of users:

1. Administrators
2. Customer
3. Tailor

2.4.1 Administrator

1. Administrators can register/login to their account.
2. Administrators can manage tailor profiles, including adding, updating, and removing tailors.
3. Administrators can oversee the overall operation of the CustomFit platform.
4. Administrators can view and manage feedback and reviews from customers.

2.4.2 Customers

1. Customers can register/login to their accounts.
2. Customers can input, store, and manage their body measurements within the app.
3. Customers can design and customize clothing items, selecting fabrics, colors, styles, and additional features.
4. Customers can track the status of their orders in real-time and manage order details, including modifications and cancellations.
5. Customers can access their order history, including past purchases and measurements.
6. Customers can provide feedback and reviews for their orders and overall experience with the app.

2.4.3 Tailors

1. Tailors can register/login to their accounts.
2. Tailors can view and manage customer orders, including order details, production status.
3. Tailors can update order status, process payments, and schedule deliveries.
4. Tailors can manage inventory, including fabric stock and production supplies.
5. Tailors can monitor customer feedback and reviews.

2.5 Operating Environment

1. Operating System: Android Hardware: Mobile Camera
2. Database: Firebase
3. Language: Python, JavaScript
4. Libraries: Tensor Flow, React Native
5. Development Environment: Android Studio.

3 Specific Requirements

3.1 Functional Requirements

These are the modules in our project.

1. Clothing Design and Customization
2. Measurement Management
3. Order Tracking and Management
4. Secure Payment Processing
5. Order History and Favorites
6. Feedback and Reviews.

3.1.1 Clothing Design and Customization:

This module enables users to design and customize their clothing items, including selecting fabrics, colors, styles, and additional features.

3.1.2 Measurement Management:

CustomFit provides tools for users to input, store, and manage their body measurements within the app, ensuring accurate and personalized fitting for each clothing order.

3.1.3 Order Tracking and Management:

Users can track the status of their orders in real-time, manage order details, make modifications, and communicate with tailors throughout the production process.

3.1.4 Secure Payment Processing:

CustomFit integrates secure payment gateways, enabling users to make payments for their orders within the app using various payment methods.

3.1.5 Order History and Favorites:

CustomFit provides users with access to their order history, including past purchases and measurements. Users can also save favorite designs or clothing configurations for future reference.

3.1.6 Feedback and Reviews:

Users can provide feedback and reviews for their orders and overall experience with the app, helping to improve service quality and provide insights for future enhancements.

3.2 Behavior Requirements

Behavior requirements describe the behavior of the system between inputs and outputs. This figure illustrates how the user interacts with the system for performing different functionalities. The behavioral requirements of our system as shown below.

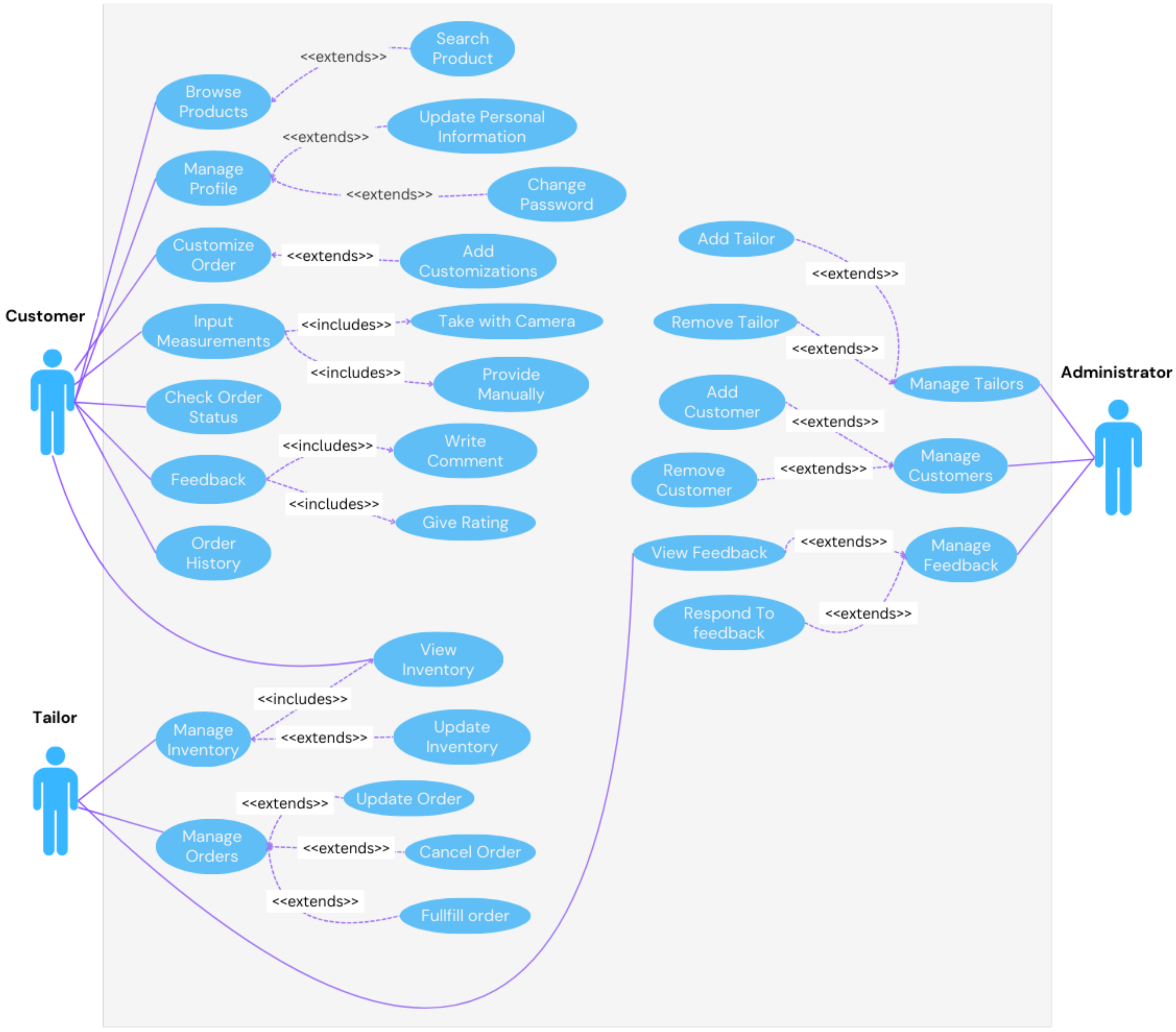

Figure 3.2- Use Case Diagram of CustomFit

3.3 External Interface Requirements

Specifies the requirements for interactions between the system and external entities.

3.3.1 User Interfaces

Following are the login screens which can be viewed by each actor in our project.

Welcome Screen

When the user launches the app, they will be presented with a welcome screen. This screen introduces the user to the app and provides navigation options to either log in or sign up

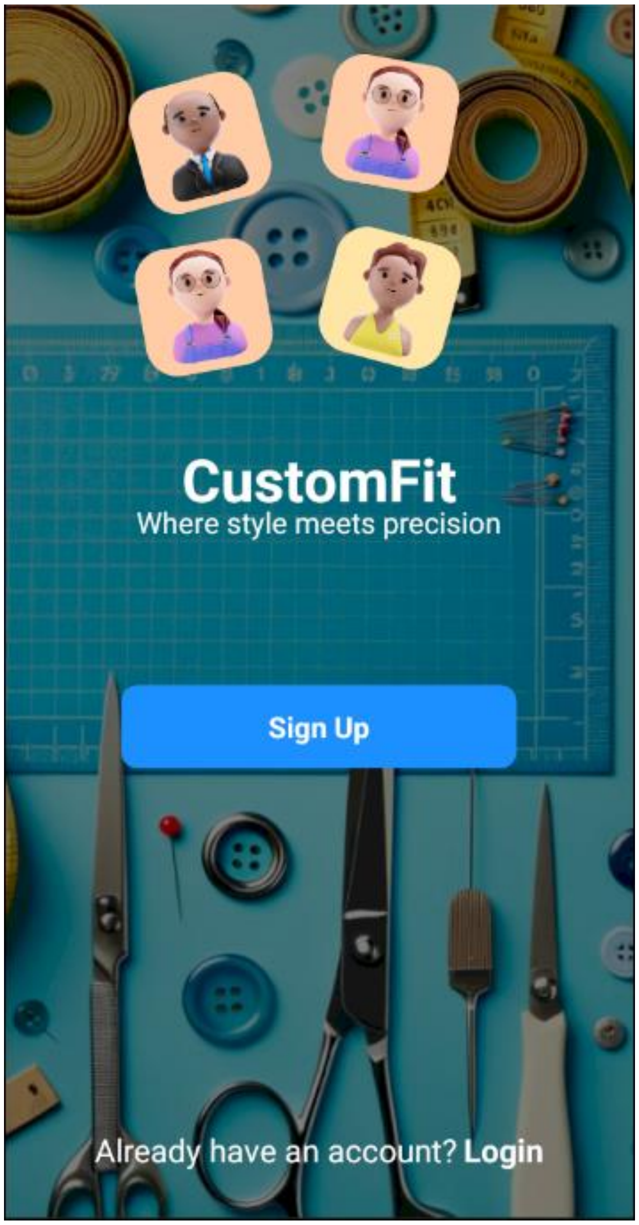

Figure 3.2 – Pre – Login Screen

Login Screen

Users can enter their email address and password to log into the app.

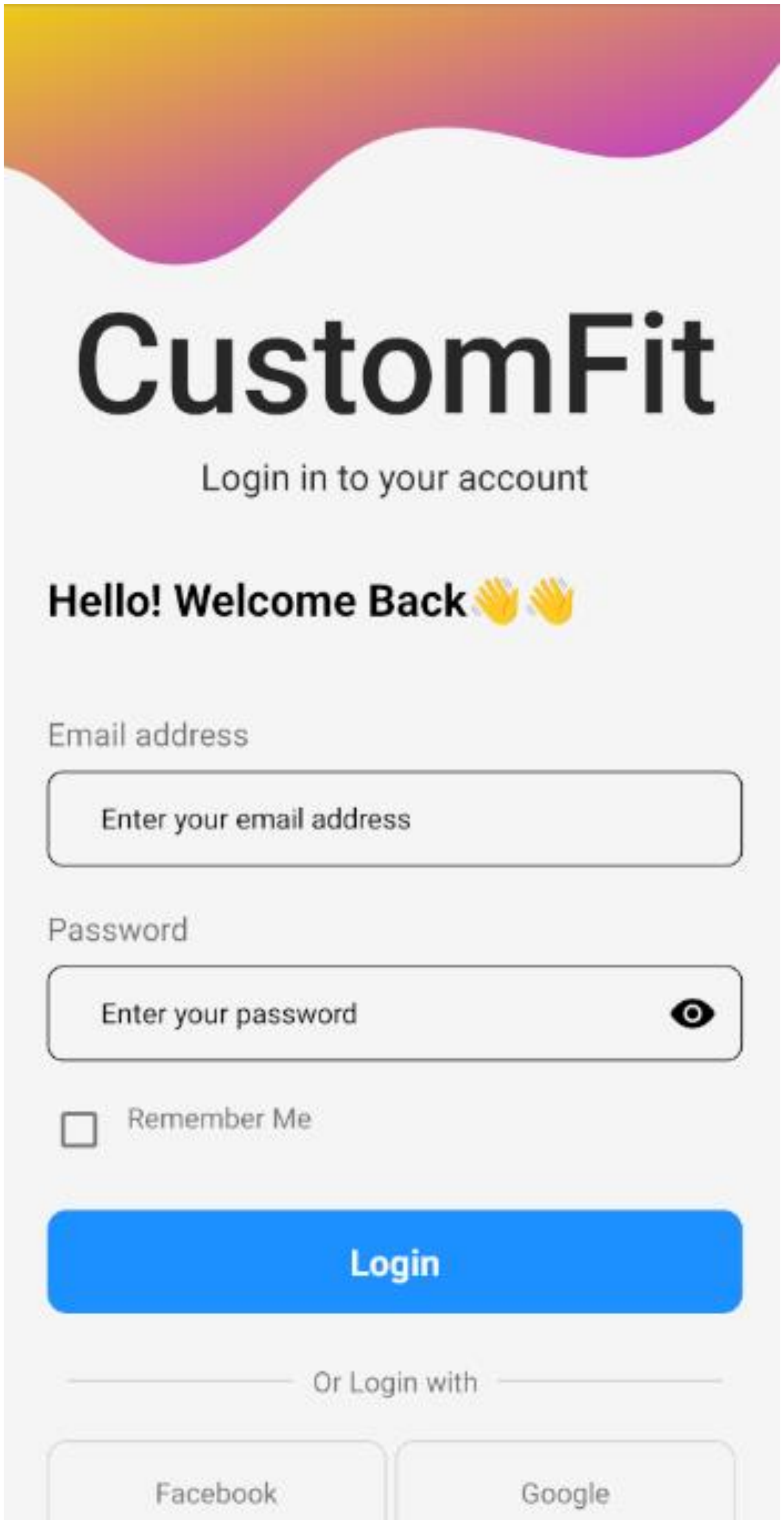

Figure 3.3 – CustomFit Login Screen

Sign Up Screen

New users can create an account by entering their email address, password, and agreeing to the terms and conditions.

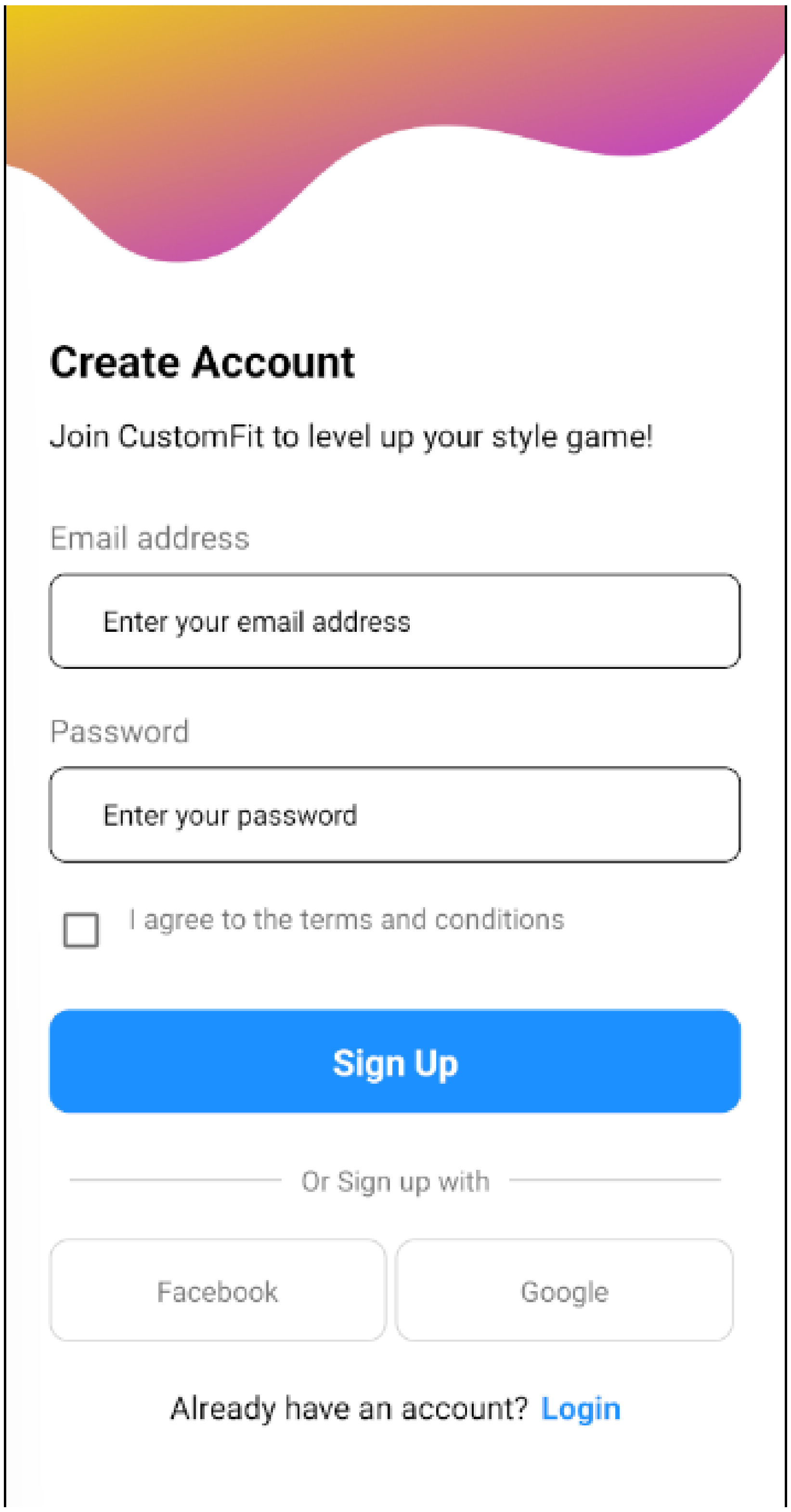

Figure 3.4 – CustomFit Sign Up Screen

Dashboard Screen

After logging in, users are directed to their dashboard. Here, they can browse different categories of clothing items.

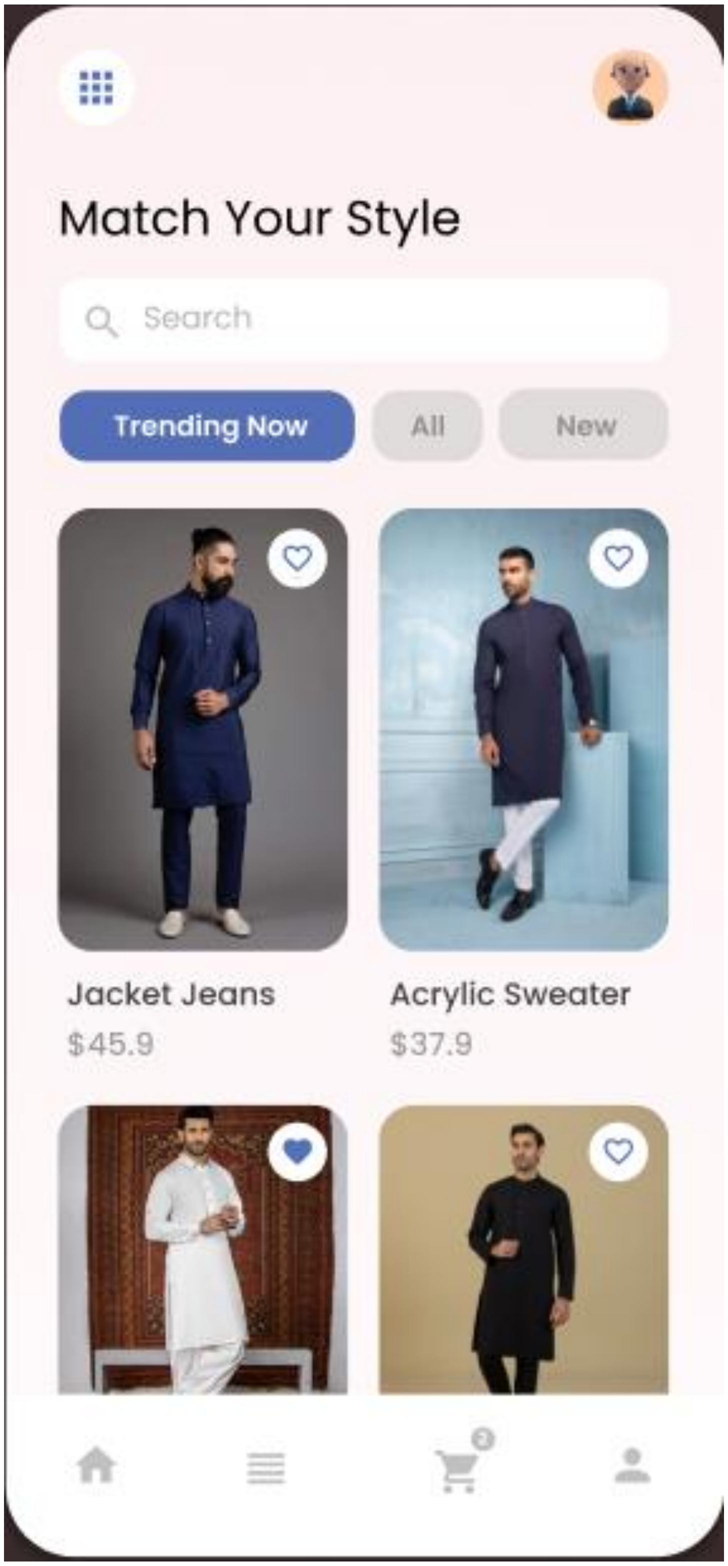

Figure 3.4 - Dashboard Screen

Product Detail Screen

When a user selects a product, they are taken to the product detail screen. This screen displays the product image, price, available sizes, color options, and an option to add the item to the cart.

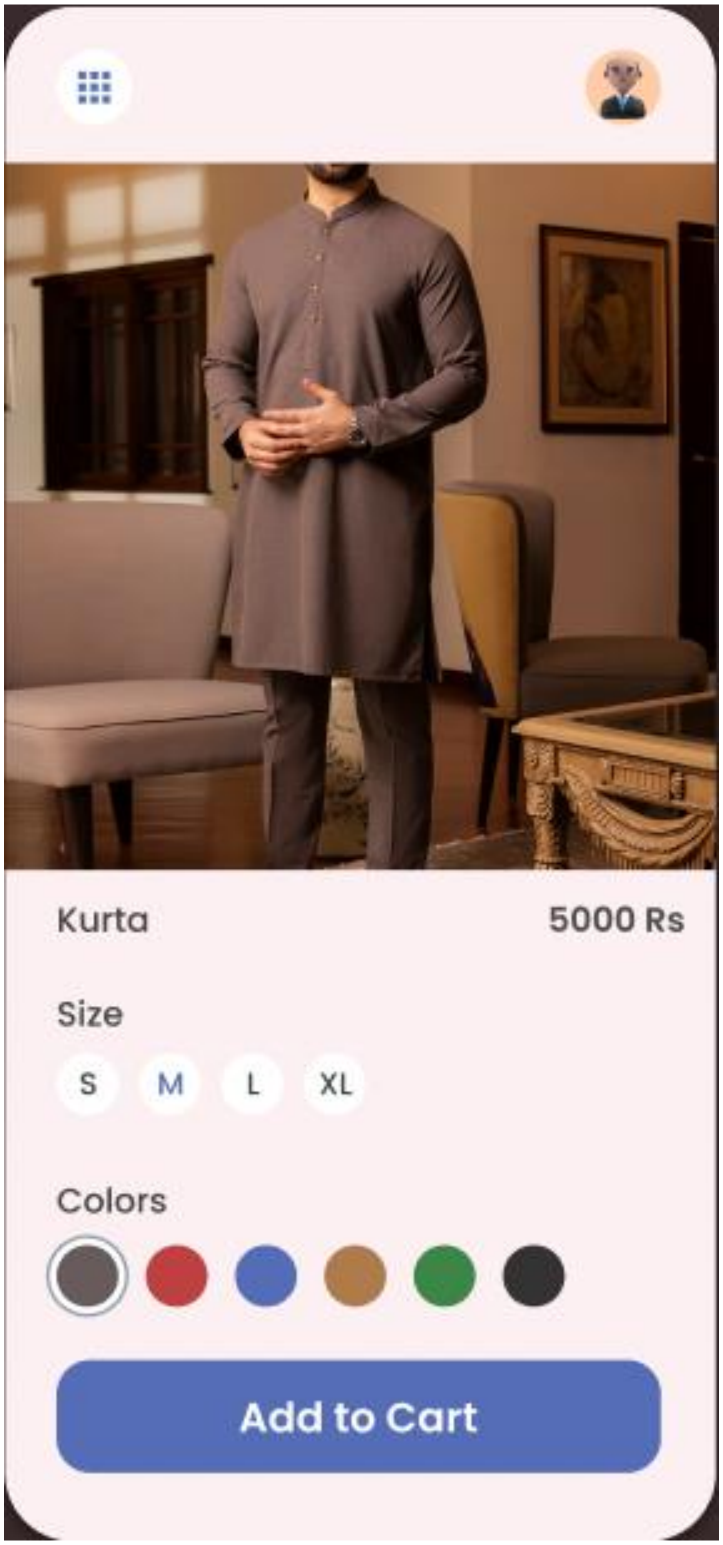

Figure 3.5 - Product Detail Screen

Cart Screen

The cart screen shows the items added to the user's cart. It includes details like product names, prices, sizes, colors, and a total cost including shipping. Users can proceed to checkout from this screen.

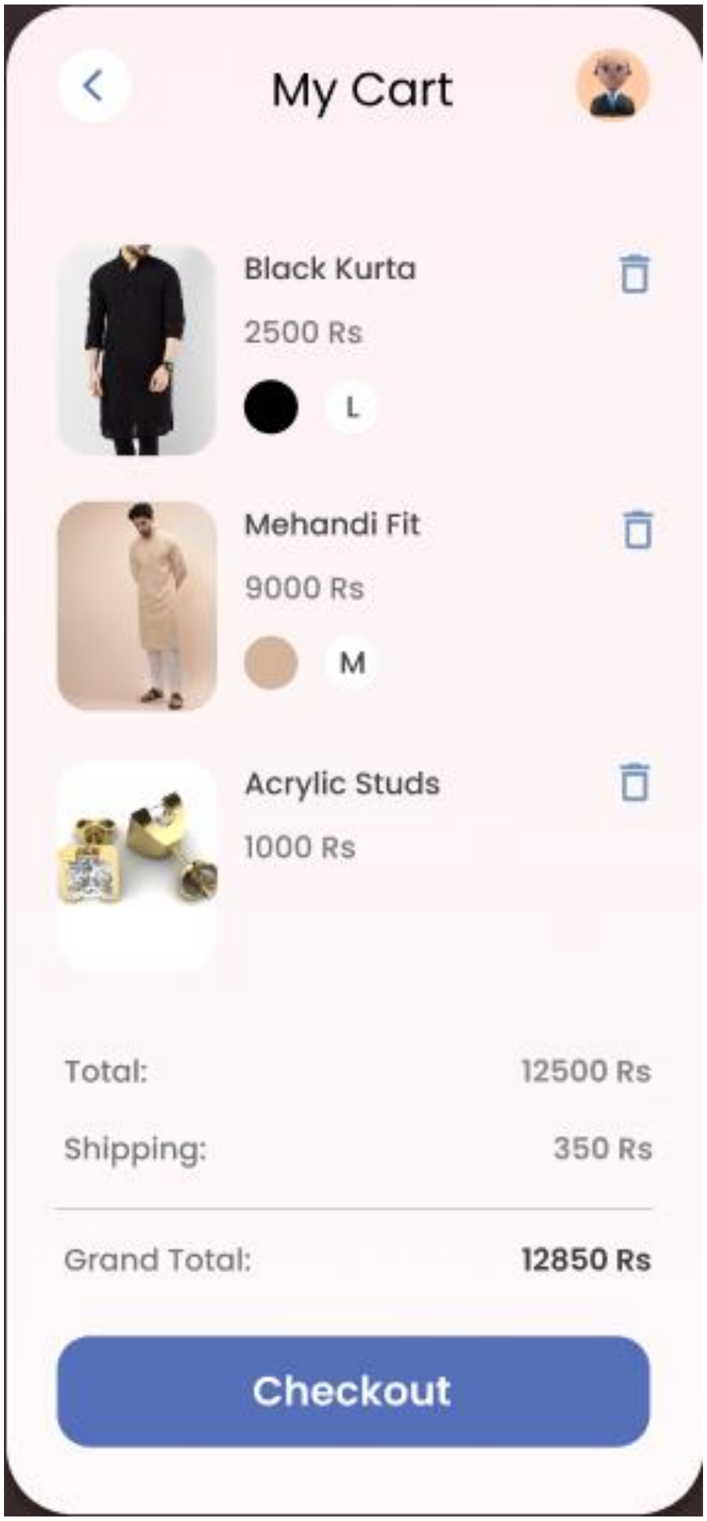

Figure 3.6 - Cart Screen

3.3.2 Hardware Interfaces

No hardware interfaces will be used in our system.

3.3.3 Software Interfaces

Following are the software interfaces which are used in our system.

Table 3.1- Software Interfaces

Tools	Version	Relation
Android Studio	*2023 above*	*Code Editor*
React Native	*0.70*	*Front End*
NodeJS	*21.7*	*Back End*
Firebase	*13.7*	*Database*
TensorFlow	*2.15*	*Machine Learning*

4 Other Non-functional Requirements

4.1 Performance Requirements

Outlines the expected performance criteria such as response times, throughput, and resource utilization for the system.

4.1.1 Response Time

The system shall not take more than 5 seconds to process the input requested from the user.

4.1.2 Availability

The system shall be available 24/7, more than 98% of the time except 2% of the time for the maintenance on the weekends.

4.1.3 Accuracy

The system shall give accurate outputs to the relevant input. The system must be free of bugs and go through a series of tests before delivery.

4.2 Safety and Security Requirements

The system shall authenticate users by making sure no one uses the wrong credentials to sign in.

4.3 Software Quality Attributes

These attributes ensure that the software meets certain standards of performance, reliability, usability, and maintainability to deliver a high-quality user experience.

4.3.1 Usability

For enhanced usability, the CustomFit system features an intuitive interface and user-friendly navigation, facilitating seamless interactions for customers, administrators, and tailors alike.

4.3.2 User Interface

The design of the system and its overall look should be appealing.

- **Ease of use**

  The system shall be easy to use for any kind of user and the user should get acquainted with the system within 5 minutes of usage.

4.3.3 Learnability

A person with a basic grasp of the language can understand. 97% of users will be able to use efficiently after 20 minutes of initial use with a 0.3% error rate.

4.3.4 Reliability

The CustomFit system prioritizes reliability by implementing robust error-handling mechanisms and conducting regular system maintenance, ensuring uninterrupted service delivery and consistent performance.

- **Availability**

  The system shall be available 24/7, more than 98% of the time except 2% of the time for the maintenance on the weekends.

- **Robustness**

  The system's recovery time should not exceed 10 minutes in 99% of cases of the system's failure.

4.3.5 Portability

The system should be able to work on different platforms like different screen sizes and render every component accurately.

5 Design Description

5.1 Composite Viewpoint

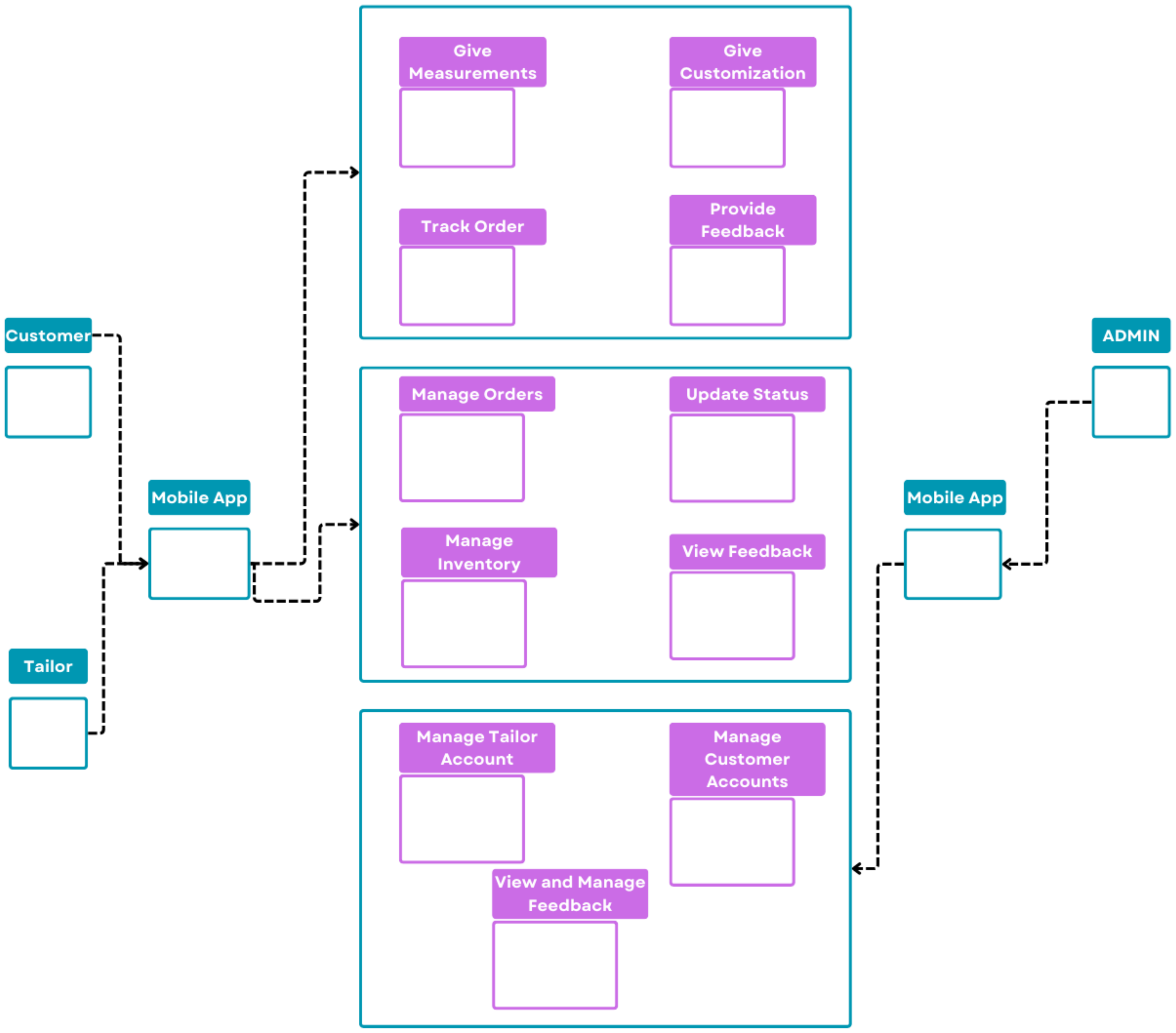

Figure 5.1– Activity Diagram CustomFit

5.2 Logical Viewpoint

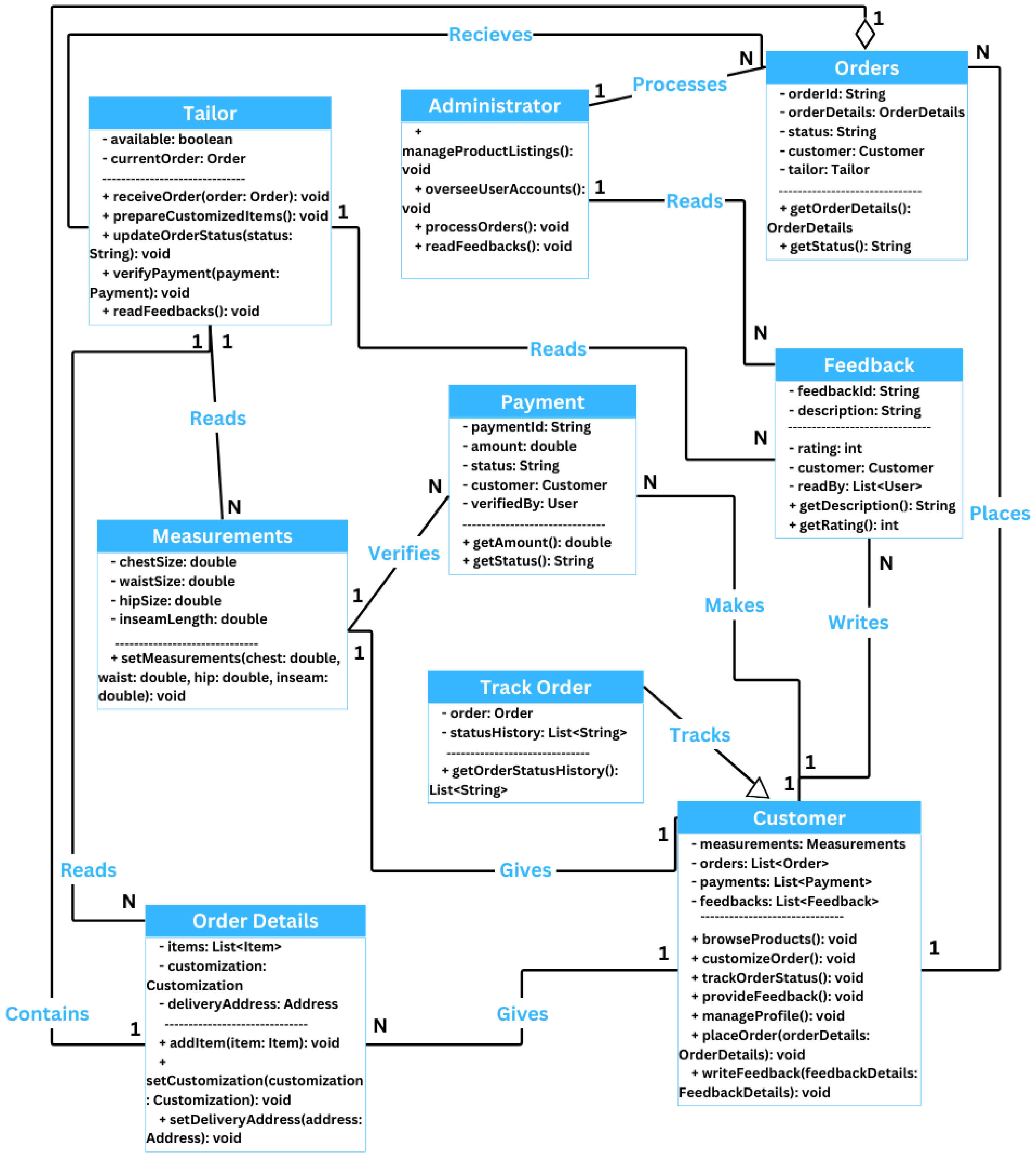
Recieves
Processes
Reads
Reads
Reads
Reads
Verifies
Makes
Writes
Places
Tracks
Gives
Gives
Contains
Tailor
- available: boolean
- currentOrder: Order
+ receiveOrder(order: Order): void
+ prepareCustomizedItems(): void
+ updateOrderStatus(status: String): void
+ verifyPayment(payment: Payment): void
+ readFeedbacks(): void
Administrator
+ manageProductListings(): void
+ overseeUserAccounts(): void
+ processOrders(): void
+ readFeedbacks(): void
Orders
- orderId: String
- orderDetails: OrderDetails
- status: String
- customer: Customer
- tailor: Tailor
+ getOrderDetails(): OrderDetails
+ getStatus(): String
Feedback
- feedbackId: String
- description: String
- rating: int
- customer: Customer
- readBy: List<User>
+ getDescription(): String
+ getRating(): int
Payment
- paymentId: String
- amount: double
- status: String
- customer: Customer
- verifiedBy: User
+ getAmount(): double
+ getStatus(): String
Measurements
- chestSize: double
- waistSize: double
- hipSize: double
- inseamLength: double
+ setMeasurements(chest: double, waist: double, hip: double, inseam: double): void
Track Order
- order: Order
- statusHistory: List<String>
+ getOrderStatusHistory(): List<String>
Customer
- measurements: Measurements
- orders: List<Order>
- payments: List<Payment>
- feedbacks: List<Feedback>
+ browseProducts(): void
+ customizeOrder(): void
+ trackOrderStatus(): void
+ provideFeedback(): void
+ manageProfile(): void
+ placeOrder(orderDetails: OrderDetails): void
+ writeFeedback(feedbackDetails: FeedbackDetails): void
Order Details
- items: List<Item>
- customization: Customization
- deliveryAddress: Address
+ addItem(item: Item): void
+ setCustomization(customization: Customization): void
+ setDeliveryAddress(address: Address): void
1
N

Figure 5.2– Class Diagram CustomFit

5.3 Information Viewpoint

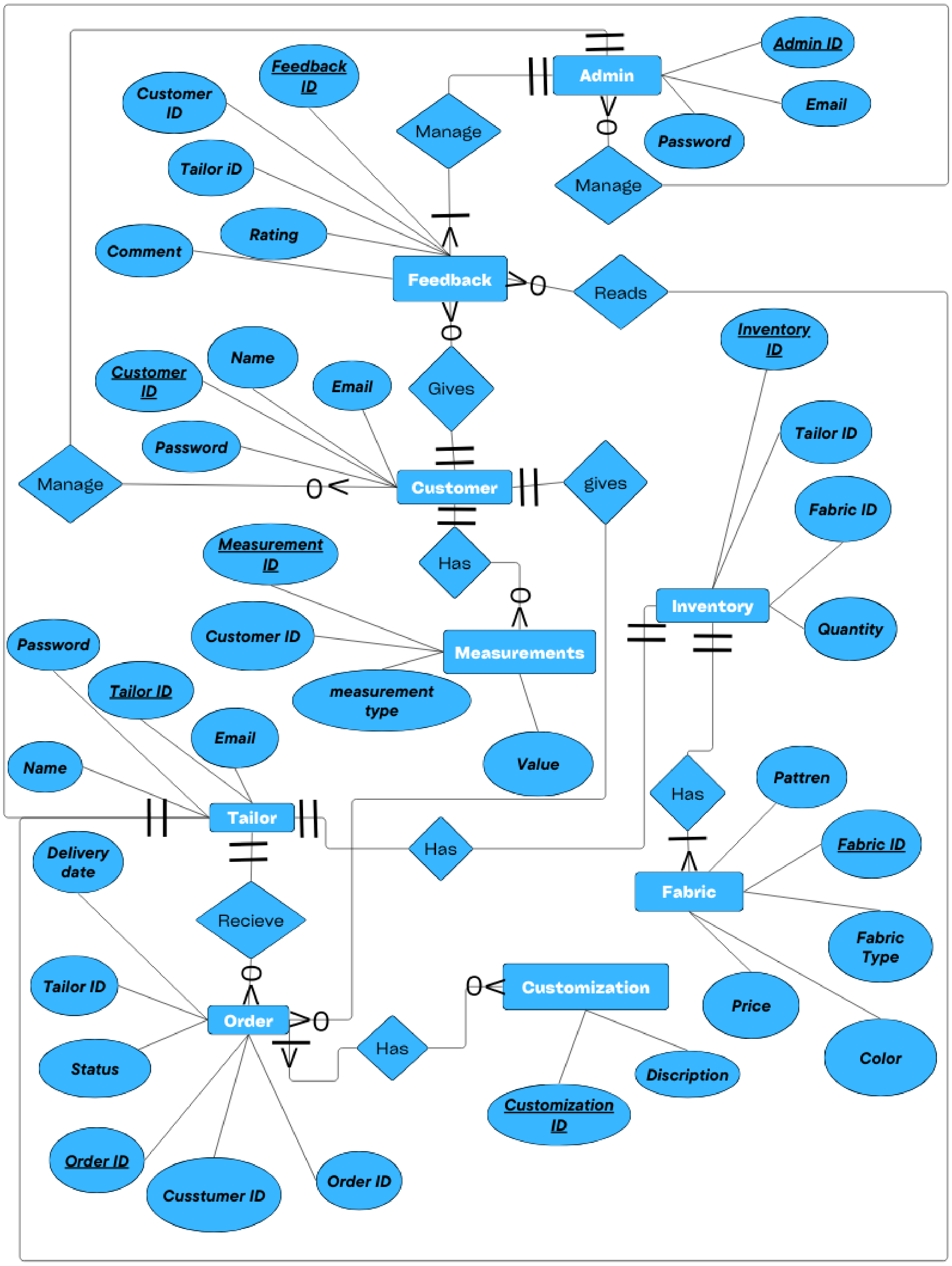

Figure 5.3 – ER Diagram CustomFit

5.4 Interaction Viewpoint

5.4.1 Customer Sequence Diagram

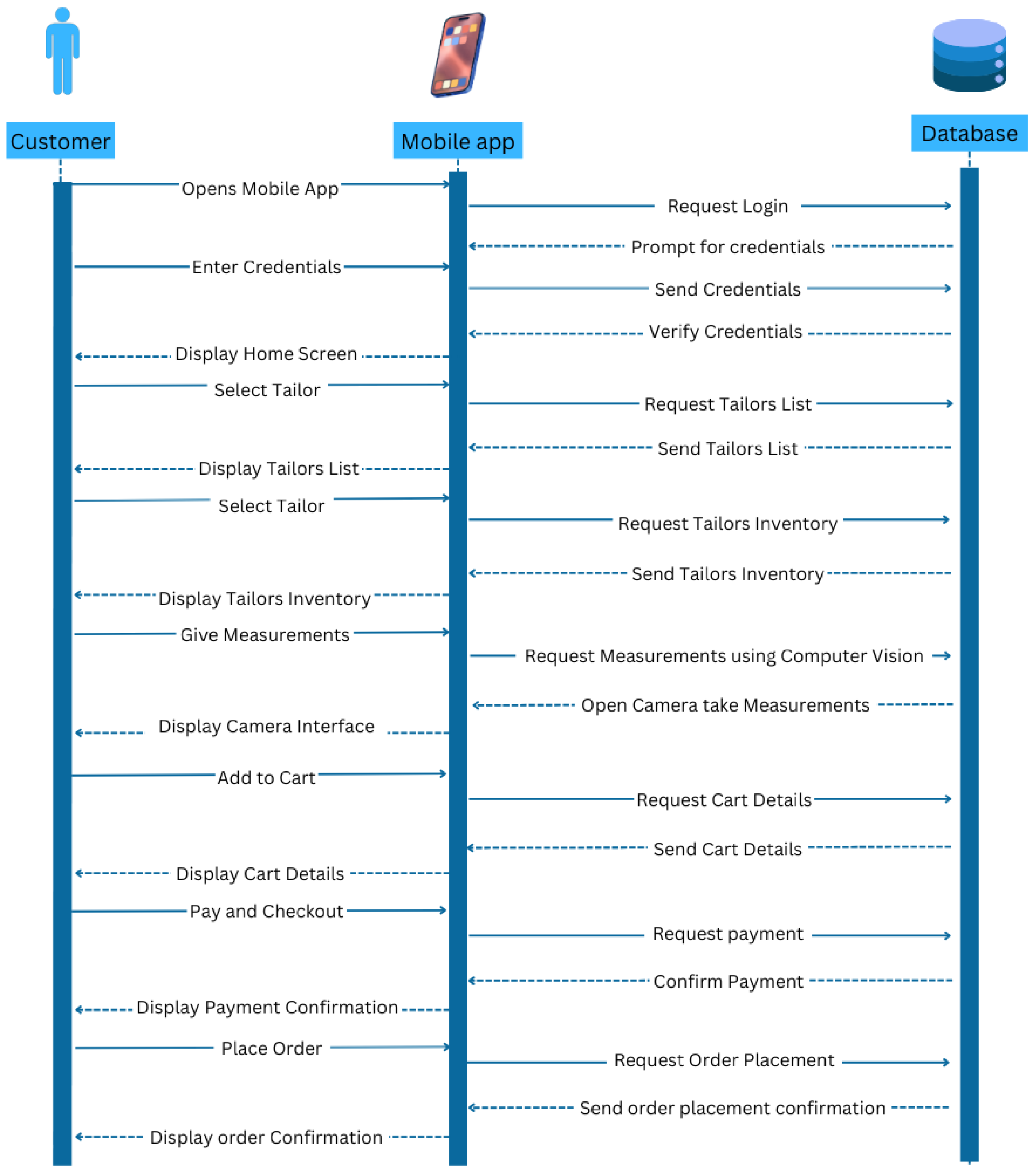

Figure 5.4 – Sequence Diagram of Customer

5.4.2 Tailor Sequence Diagram

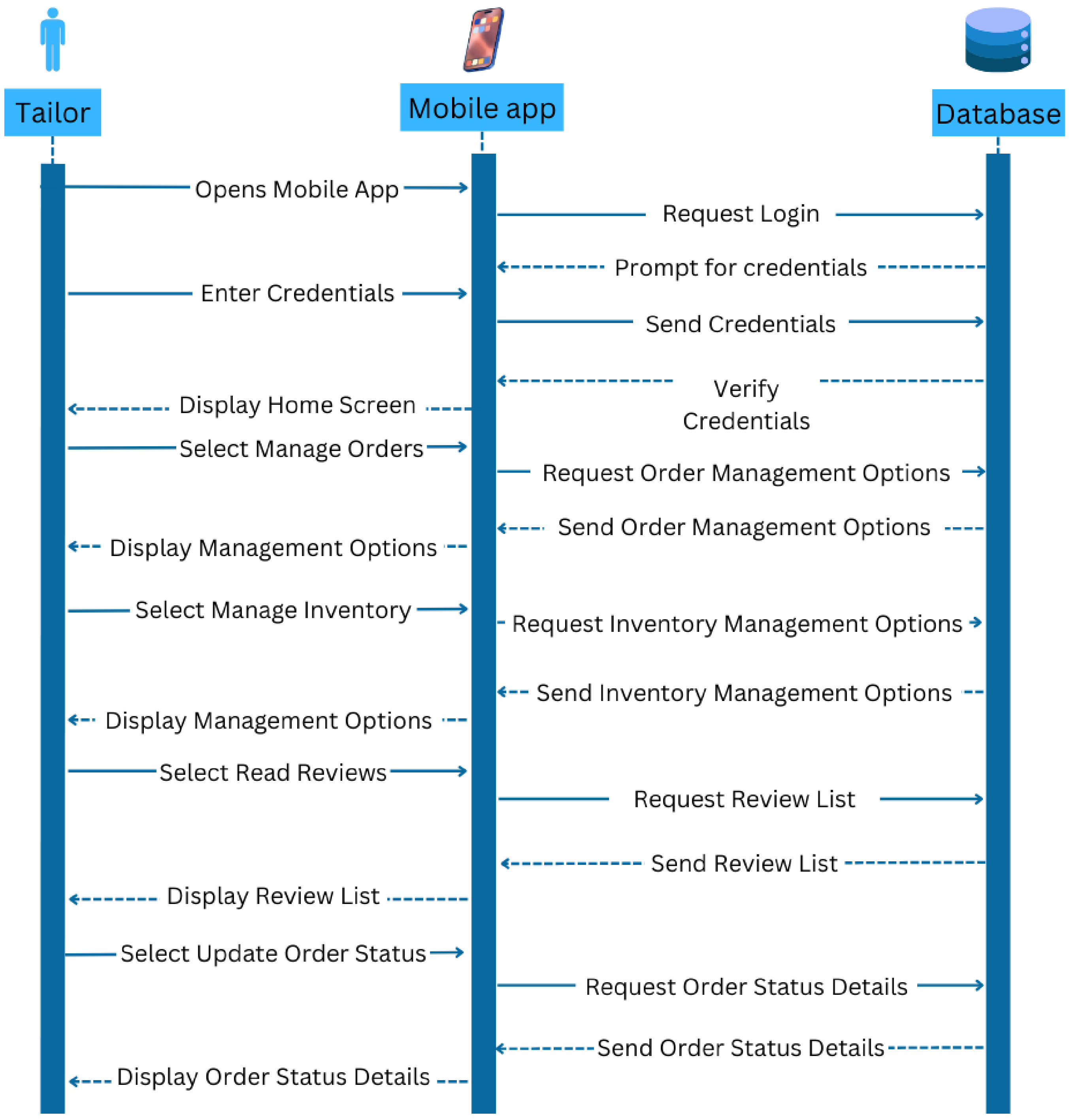
Tailor
Mobile app
Database
Opens Mobile App
Request Login
Prompt for credentials
Enter Credentials
Send Credentials
Verify Credentials
Display Home Screen
Select Manage Orders
Request Order Management Options
Send Order Management Options
Display Management Options
Select Manage Inventory
Request Inventory Management Options
Send Inventory Management Options
Display Management Options
Select Read Reviews
Request Review List
Send Review List
Display Review List
Select Update Order Status
Request Order Status Details
Send Order Status Details
Display Order Status Details

Figure 5.5 – Sequence Diagram Tailor

5.4.3 Administrator Diagram

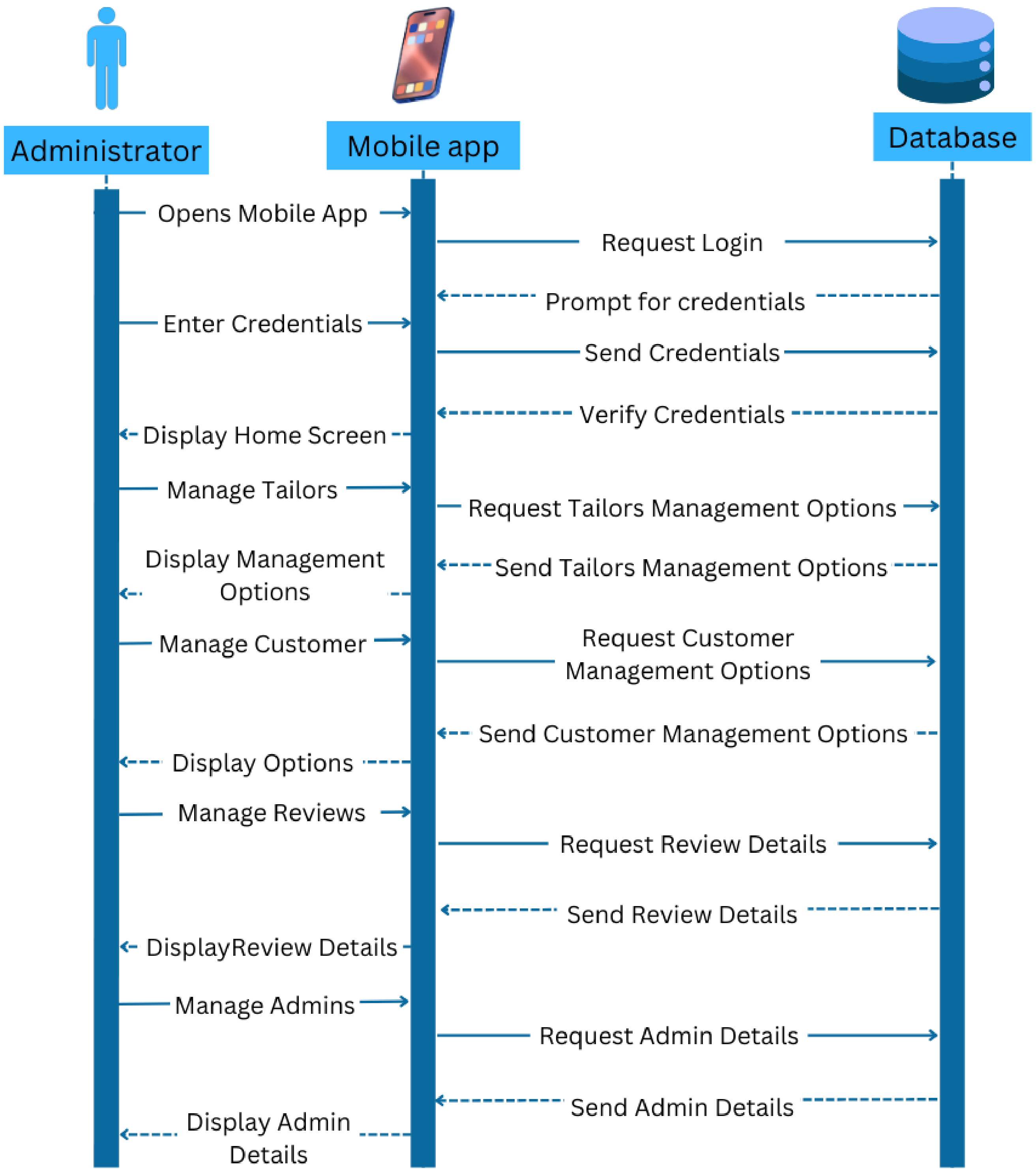

Figure 5.6 – Sequence Diagram administrator

5.5 State Dynamics Viewpoint

5.5.1 State Dynamic of the Customer

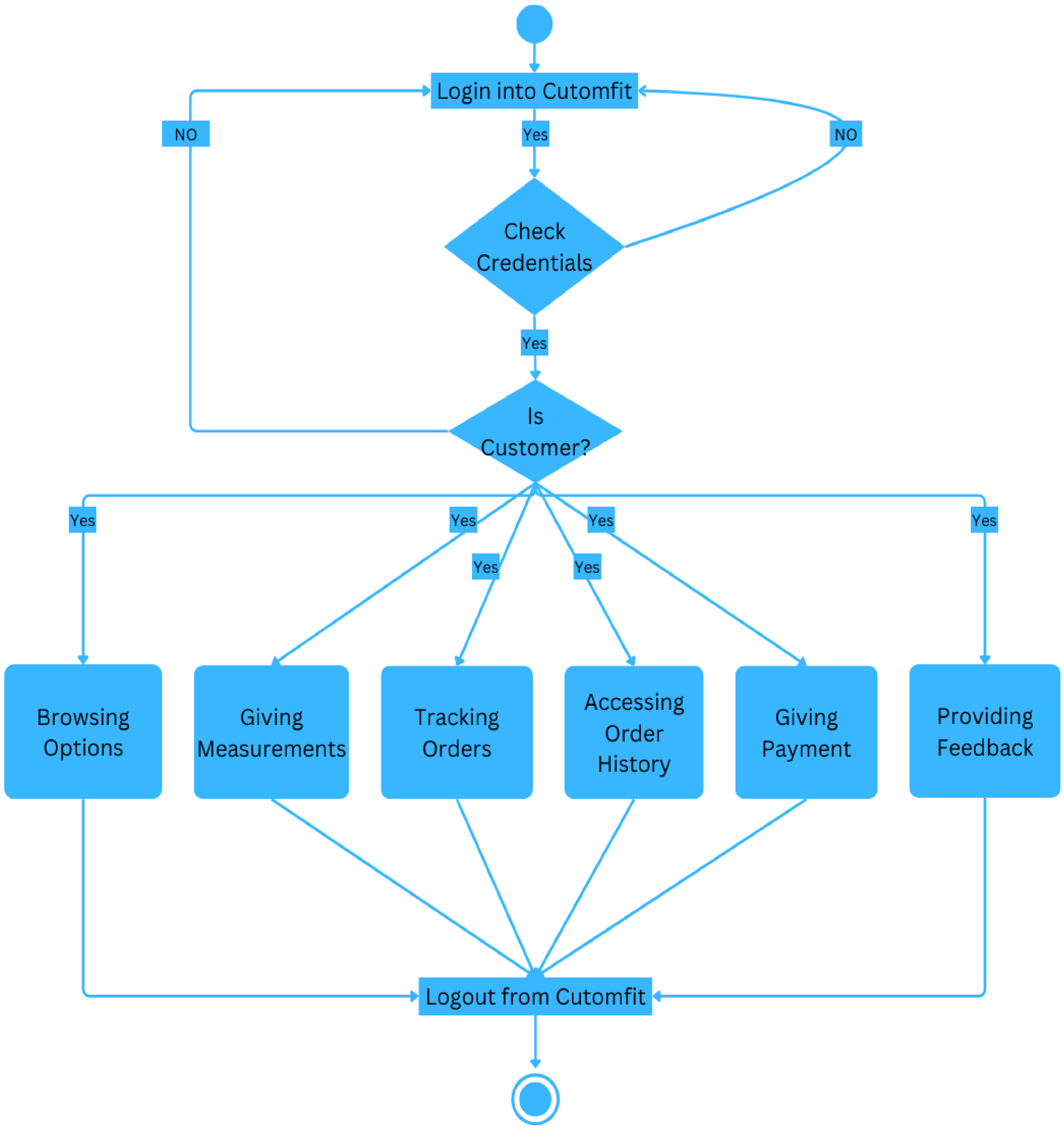

Figure 5.7 – State Dynamic of Customer

5.5.2 State Dynamic of Tailor

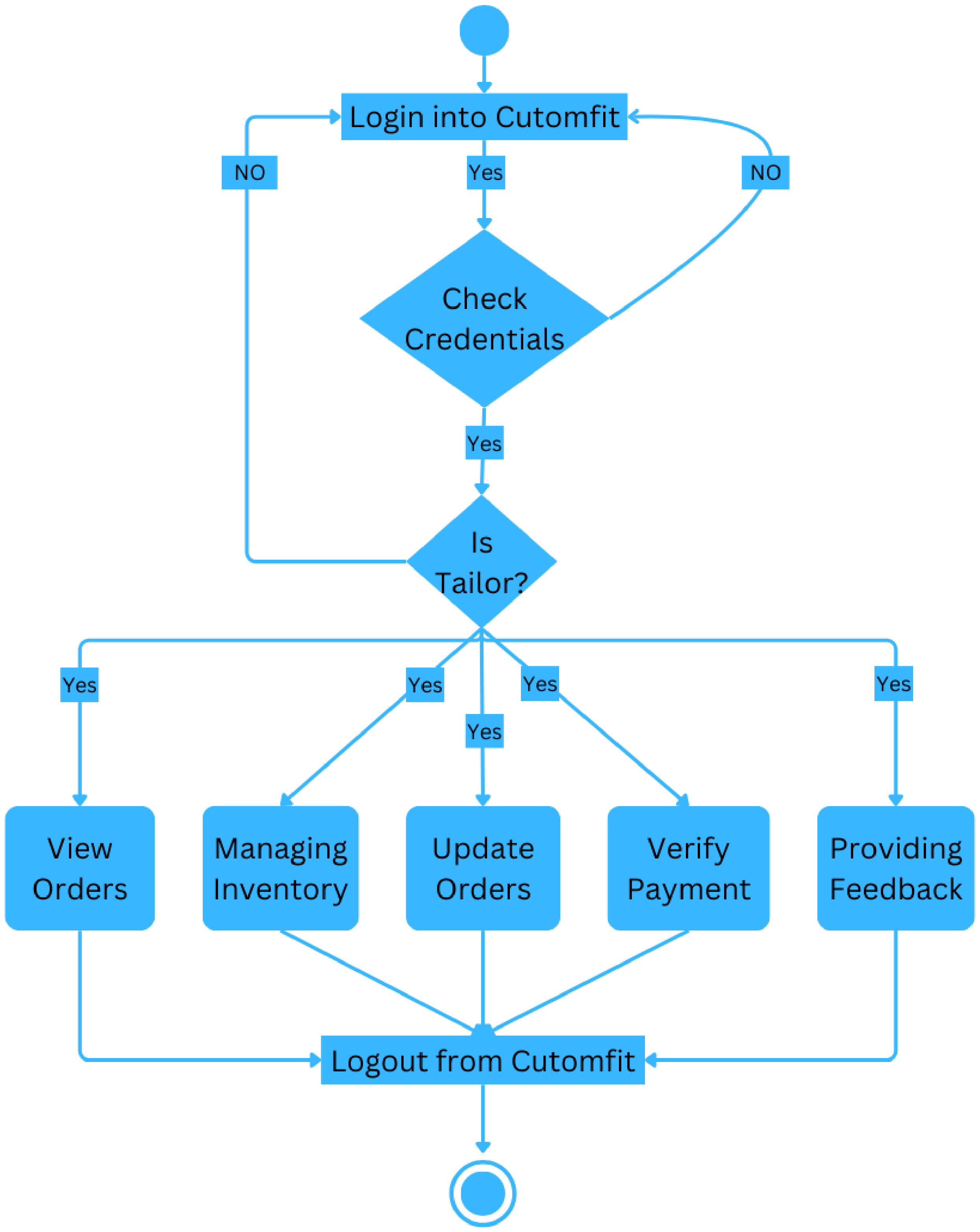

Figure 5.8 – State Dynamic of Tailor

5.5.3 State Dynamic of Administrator

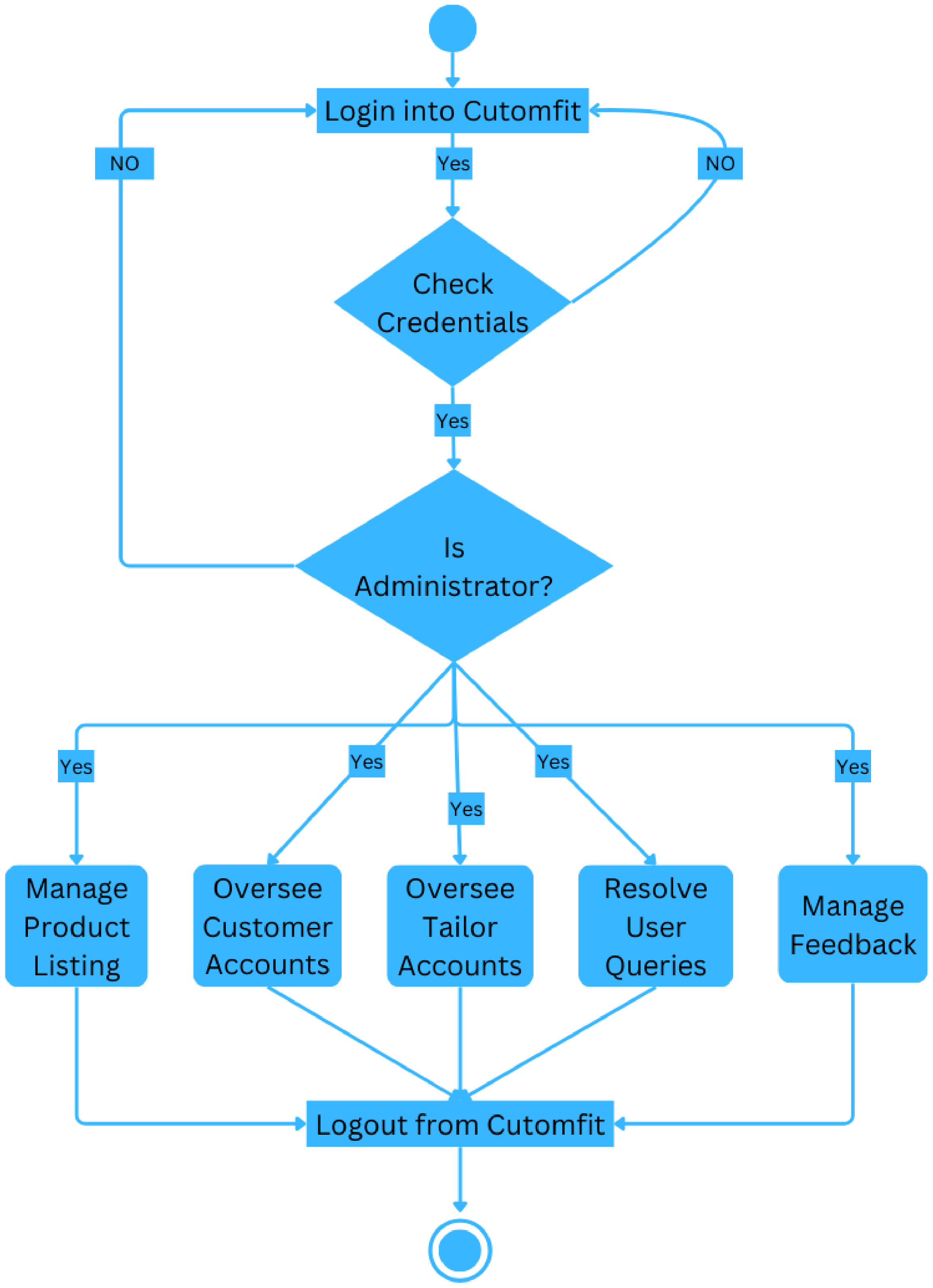

Figure 5.9 – State Dynamic of Administrator

5.6 Algorithm Viewpoint

5.6.1 Algorithmic Viewpoint of Login

Table 5.1-Algorithmic Viewpoint of Login

	Login
Description: This class Describes the responsibilities and functionalities of User authentication. Users can log in to perform further action by using this class.	
Attributes	**Description**
Employee ID	The employee id will be unique and a differentiating attribute for user
Password	The password would be stored hashed. The user would only be able to access the accounts only if he/she enters the correct password.
Methods	**Description**
Login ()	**Algorithm**
	1. Get the username and password from the user, from the input fields. 2. Match both password and email with the email and password in the Database. 3. If one of these is incorrect Show the error message: “Invalid Credentials” 4. Else If both matches show the success 5. code and message: “Successfully Login” and navigate to the dashboard page
	Pseudo Code
	START DEFINE function handleLogin(email, password) DISPLAY loading indicator

```
    IF user DOES NOT EXIST
        DISPLAY error message: "Invalid Credentials"
        RETURN
    IF user.password EQUALS input password
        DISPLAY success message: "Successfully Login"
        NAVIGATE to Dashboard
    ELSE
        DISPLAY error message: "Invalid Credentials"
END FUNCTION
ON loginButton CLICK
    GET email from email input field
    GET password from password input field
    CALL handleLogin(email, password)
END
```

5.6.2 Algorithmic Viewpoint of Customer

Table 5.2– Algorithmic Viewpoint of Customer

	Customer Resources
Description: This class describes the characteristics and functionalities of Customer resources.	
Attributes	**Description**
Name	The name of the Customer resource is stored in the attribute
Methods	**Description**
Browse_products()	A method is used to browse products
	Algorithm

1. Click on browse products.
2. Show available products.
3. If user selects a product.
4. Display product details.
5. Else.
6. Show message "No products available".

Pseudo Code

```
START
DEFINE function browseProducts()
    DISPLAY loading indicator
    products = GET all products from database
    IF products IS EMPTY
        DISPLAY message: "No products available"
        HIDE loading indicator
        RETURN
    DISPLAY products list
    ON product selection
        selectedProduct = GET details of the selected product from products list
        IF selectedProduct EXISTS
            DISPLAY selectedProduct details
        ELSE
            DISPLAY message: "Product not found"
    HIDE loading indicator
END FUNCTION
ON browseProductsButton CLICK
```

	CALL browseProducts() END
Customize_order()	A method is used to customize an order
	Algorithm
	1. Click on customize order. 2. Show customization options. 3. If user customizes the order. 4. Apply customization. 5. Else. 6. Show message "Order customization canceled".
	Pseudo Code

	START
	DEFINE function customizeOrder()
	  DISPLAY customization options
	  ON customization options submission
	    IF customization options are provided
	      APPLY customization to the order
	      DISPLAY message: "Customization applied successfully"
	    ELSE
	      DISPLAY message: "Order customization canceled"
	END FUNCTION
	ON customizeOrderButton CLICK
	  CALL customizeOrder()
	END

Methods	Description
Write_review()	A method used to write a review
	Algorithm
	1. Click on write review. 2. Show review form. 3. If user submits the review. 4. Save the review. 5. Else. 6. Show message "Review canceled".
	Pseudo Code

	START
	DEFINE function writeReview()
	    DISPLAY review form
	    ON review form submission
	        IF review form data is provided
	            SAVE the review data to the database
	            DISPLAY message: "Review submitted successfully"
	        ELSE
	            DISPLAY message: "Review canceled"
	END FUNCTION
	ON writeReviewButton CLICK
	    CALL writeReview()
	END

Select_tailor()	A method used to select a tailor
	Algorithm

1. Click on select tailor.
2. Show list of available tailors.
3. If user selects a tailor.
4. Assign order to selected tailor.
5. Else.
6. Show message "No tailors available".

Pseudo Code

```
START
DEFINE function selectTailor()
    DISPLAY loading indicator
    tailors = GET all available tailors from database
    IF tailors IS EMPTY
        DISPLAY message: "No tailors available"
        HIDE loading indicator
        RETURN
    DISPLAY tailors list
    ON tailor selection
        selectedTailor = GET details of the selected tailor from tailors list
        IF selectedTailor EXISTS
            ASSIGN order to selected tailor
            DISPLAY message: "Order assigned to selected tailor"
        ELSE
            DISPLAY message: "Tailor not found"
    HIDE loading indicator
```

```
END FUNCTION

ON selectTailorButton CLICK
    CALL selectTailor()
END
```

See_tailor_inventory()	A method used to see tailor's inventory

Algorithm

1. Click on see tailor's inventory.
2. Show tailor's inventory.
3. If user wants to order from inventory.
4. Add items to the order.
5. Else.
6. Show message "No items available in tailor's inventory".

Pseudo Code

```
START
DEFINE function seeTailorInventory()
    DISPLAY loading indicator
    inventory = GET tailor's inventory from database
    IF inventory IS EMPTY
        DISPLAY message: "No items available in tailor's inventory"
        HIDE loading indicator
        RETURN
    DISPLAY inventory list
    ON item selection
        IF item is selected
```

```
            ADD selected items to the order
            DISPLAY message: "Items added to the order"
        ELSE
            DISPLAY message: "No items selected"
    HIDE loading indicator
END FUNCTION
ON seeTailorInventoryButton CLICK
    CALL seeTailorInventory()
END
```

Make_payment()	A method used to make a payment

Algorithm

1. Click on make payment
2. Enter payment details
3. If payment is successful
4. Complete the order
5. Else
6. Show message "Payment failed"

Pseudo Code

```
START
DEFINE function makePayment()
    DISPLAY payment form
    ON payment form submission
        paymentDetails = GET payment details from form
        paymentResult = PROCESS payment with paymentDetails
        IF paymentResult IS SUCCESSFUL
```

```
            COMPLETE the order
            DISPLAY message: "Payment successful. Order completed"
        ELSE
            DISPLAY message: "Payment failed"
END FUNCTION
ON makePaymentButton CLICK
    CALL makePayment()
END
```

5.6.3 Algorithmic Viewpoint of Tailor

Table 5.3 – Algorithmic Viewpoint of Tailor

	Tailor Resources
Description: This class describes the characteristics and functionalities of Tailor resources.	
Attributes	**Description**
Name	The name of the Tailor resource is stored in the attribute
Methods	**Description**
Receive_order()	A method is used to receive an order
	Algorithm
	1. Receive order details. 2. If order received successfully. 3. Process the order. 4. Else. 5. Show message "Order reception failed".

Pseudo Code

```
START
DEFINE function receiveOrder(orderDetails)
   receivedOrder = RECEIVE orderDetails
   IF receivedOrder IS SUCCESSFUL
      PROCESS order
      DISPLAY message: "Order received and processing"
   ELSE
      DISPLAY message: "Order reception failed"
END FUNCTION
ON orderDetailsReceived(orderDetails)
   CALL receiveOrder(orderDetails)
END
```

Prepare_customized_items()	A method used to prepare customized items

Algorithm

1. Receive customization details.
2. Prepare customized items.
3. If items prepared successfully.
4. Notify customer.
5. Else.
6. Show message "Item preparation failed".

Pseudo Code

```
START
DEFINE function prepareCustomizedItems(customizationDetails)
    receivedCustomization = RECEIVE customizationDetails
    preparationResult = PREPARE items with receivedCustomization
    IF preparationResult IS SUCCESSFUL
        NOTIFY customer about successful preparation
        DISPLAY message: "Customized items prepared successfully"
    ELSE
        DISPLAY message: "Item preparation failed"
END FUNCTION
ON customizationDetailsReceived(customizationDetails)
    CALL prepareCustomizedItems(customizationDetails)
END
```

Update_order_status()	A method used to update the order status

Algorithm

1. Receive updated order status
2. Update order status in the system
3. If status updated successfully
4. Notify customer
5. Else
6. Show message "Order status update failed"

Pseudo Code

```
START
DEFINE function updateOrderStatus(orderStatusDetails)
    receivedStatus = RECEIVE orderStatusDetails
    updateResult = UPDATE order status in the system with receivedStatus
    IF updateResult IS SUCCESSFUL
        NOTIFY customer about successful status update
        DISPLAY message: "Order status updated successfully"
    ELSE
        DISPLAY message: "Order status update failed"
END FUNCTION
ON orderStatusDetailsReceived(orderStatusDetails)
    CALL updateOrderStatus(orderStatusDetails)
END
```

Verify_payment()	A method used to verify a payment

Algorithm

1. Receive payment details.
2. Verify payment status.
3. If payment verified successfully.
4. Proceed with order processing.
5. Else.
6. Show message "Payment verification failed".

Pseudo Code

```
START
DEFINE function verifyPayment(paymentDetails)
```

```
    receivedPayment = RECEIVE payment details
    verificationResult = VERIFY payment status with receivedPayment
    IF verificationResult IS SUCCESSFUL
        PROCESS the order
        DISPLAY message: "Payment verified successfully. Order processing"
    ELSE
        DISPLAY message: "Payment verification failed"
END FUNCTION
ON paymentDetailsReceived(paymentDetails)
    CALL verifyPayment(paymentDetails)
END
```

Read_feedbacks()	A method used to read feedback
	Algorithm
	1. Access feedback from the system 2. Display feedback 3. If feedback available 4. Read and analyze feedback 5. Else 6. Show message "No feedback available"
	Pseudo Code

```
START
DEFINE function accessFeedbacks()
   feedback = GET feedback from the system
   DISPLAY feedback
   IF feedback IS NOT EMPTY
      READ and ANALYZE feedback
   ELSE
      DISPLAY message: "No feedback available"
END FUNCTION
ON feedbacksRequested()
   CALL accessFeedbacks()
END
```

5.6.4 Algorithmic Viewpoint of Administrator

Table 2.4- Algorithmic Viewpoint of Administrator

	Administrator Resources
Description: This class describes the characteristics and functionalities of Administrator resources.	
Attributes	**Description**
Name	The name of the Administrator resource is stored in the attribute.
Methods	**Description**
Manage_product_listings()	A method is used to manage product listings
	Algorithm

1. Access product listings.
2. Add, update, or remove products.
3. If product listings managed successfully.
4. Update product information in the system.
5. Else.
6. Show message "Product listing management failed".

Pseudo Code

```
START
DEFINE function manageProductListings(action, productDetails)
    productList = GET product listings from the system
    IF action IS "add"
        result = ADD product to productList with productDetails
    ELSE IF action IS "update"
        result = UPDATE product in productList with productDetails
    ELSE IF action IS "remove"
        result = REMOVE product from productList with productDetails
    IF result IS SUCCESSFUL
        UPDATE product information in the system
        DISPLAY message: "Product information updated successfully"
    ELSE
        DISPLAY message: "Product listing management failed"
END FUNCTION
```

	ON productListingAction(action, productDetails)
	   CALL manageProductListings(action, productDetails)
	END

Oversee_user_accounts()	A method used to oversee user accounts
	Algorithm
	1. Access user account information. 2. View user details and activities. 3. If user accounts are overseen successfully. 4. Analyze user data for insights. 5. Else. 6. Show message "User account oversight failed".
	Pseudo Code
	START DEFINE function overseeUserAccounts() userAccounts = GET user account information from the system DISPLAY user details and activities from userAccounts IF userAccounts IS NOT EMPTY ANALYZE user data from userAccounts DISPLAY message: "User data analyzed for insights" ELSE DISPLAY message: "User account oversight failed" END FUNCTION ON userAccountOversight() CALL overseeUserAccounts()

	END
Manage_inventory()	A method used to manage inventory
	Algorithm
	1. Access inventory information. 2. Add, update, or remove inventory items. 3. If inventory managed successfully. 4. Update inventory status in the system. 5. Else. 6. Show message "Inventory management failed".
	Pseudo Code

	START
	DEFINE function manageInventory(action, inventoryDetails)
	  inventoryList = GET inventory information from the system
	action
	  IF action IS "add"
	    result = ADD item to inventoryList with inventoryDetails
	  ELSE IF action IS "update"
	    result = UPDATE item in inventoryList with inventoryDetails
	  ELSE IF action IS "remove"
	    result = REMOVE item from inventoryList with inventoryDetails
	  IF result IS SUCCESSFUL

```
        UPDATE inventory status in the system
        DISPLAY message: "Inventory status updated successfully"
    ELSE
        DISPLAY message: "Inventory management failed"
END FUNCTION
ON inventoryAction(action, inventoryDetails)
    CALL manageInventory(action, inventoryDetails)
END
```

Read_feedbacks()	A method used to read feedback
	Algorithm

1. Access feedback from the system.
2. Display feedback.
3. If feedback available.
4. Read and analyze feedback.
5. Else.
6. Show message "No feedback available".

Pseudo Code

```
START
DEFINE function accessFeedback()
    feedback = GET feedback from the system
    DISPLAY feedback
    IF feedback IS NOT EMPTY
        READ and ANALYZE feedback
    ELSE
    DISPLAY message: "No feedback available"

END FUNCTION

ON feedbackAccess()

  CALL accessFeedback()

END
```