1. Introduction

1.1 Purpose of Document

This is a Requirements Specification document for a new eCommerce web-based app Little Tags. (LTG). LTG is a web-based e-commerce app primarily for fashion retail. This document describes the scope, objectives, and goals of the LTG. In addition to describing non-functional requirements, this document models the functional requirements with use cases, interaction diagrams, and class models. This document is intended to direct the design and implementation of the target system in an object-oriented language.

1.2 Project Summary

Project Name: Little Tags Fashion

Mentor: Gaurav Roy

Developers: Ajinkya Bodade and Vatsal Patel

1.3 Background

LTG sells state-of-the-art eCommerce app for fashion retail. LTG customers include individuals interested in fast fashion and fashion-forward clothing and high craftsmanship clothes into their wardrobe. LTG has identified two trends that they believe will cause explosive growth in the demand for their products. The first is the customers don't want to pay high retail on their clothes. The second: brick and mortar stores are a thing of the past and no one wants to leave the comfort of their couches to buy and tailor their outfits.

1.4 Project Scope

This project's scope is a web-based system that supports the marketing of LTG products directly to customers without the need for brick-and-mortar stores. Advertising of products, inventory control, and account billing are not part of this project.

1.5 System Purpose

1.5.1 Users

Those who will primarily benefit from the new system and those who will be affected :

Customers:
Customers will find site navigation, product identification, and product ordering easier. Customers can easily search for the item they're trying to purchase and can pay via their credit cards with the implementation of the STRIPE payment gateway, thereby making the payments more secure and fast. Negating the need for logistical control post ordering

1.5.2 Location

The system will be available to any potential customer using the Internet.

1.5.3 Responsibilities

The primary responsibilities of the new system:

- provide customers direct access to the up-to-date, accurate product information on which they can decide to buy
- customize product offerings to specific users
- allow differential access to web pages based on the type of user
- allow customers to place an order through the website
- allow customers to request the assistance of a sales agent
- send order information directly to Accounting and Shipping

Other desired features of the new system:

- a consistent "look and feel" throughout the website
- full-text searches of the web pages a user has permission to access
- on-line help in website navigation
- password protection scheme for non-public web pages
- translation of a web page to another language

1.6 Overview of Document

The rest of this document gives the detailed specifications for the Little Tags Fashion App. It is organized as follows:

- Section 2: Functional Objectives
  Each objective gives the desired behavior for the system, a business justification, and a measure to determine if the final system has successfully met the objective. These objectives are organized by priority. For the Little Tags Fashion App to be considered successful, all high-priority objectives must be met.

- Section 3: Technical Objectives
  This section is organized by category. Each objective specifies a technical requirement or constraint on the overall characteristics of the system. Each objective is measurable.

2. Functional Objectives

2.1 High Priority

1. LTG shall allow the user to log in/ Register into the app using either the email/password, Google/Facebook authentication. This will be achieved via the powers of the Google Firebase coupled with our custom made authentication protocols

2. LTG shall reflect a new and changed product description as soon as the product has been added to the inventory and therefore to the database, which will be achieved with the MongoDb's NoSQL database and also using the firebase's database protocols called the Firestore which will make the app a low-latent one and therefore making it real-time as much as possible
3. LTG shall display information that is customized based on the user's details (viz the name, email, saved addresses, contact info). This feature will be achieved via storing the details directly on the client-side, by using the REDUX store for content management on the client Side.
4. LTG shall allow users to view the hottest trending product on the homepage itself, thereby increasing the SEO of the website and making the UX better.
5. LTG shall allow a customer to directly categorize the desired product as per the parameters set by the user.
6. LTG shall provide lean, fast search functionality for the user, to search via the search term. LTG shall provide the user the power to add the items to both carts
7. LTG shall provide shipping with accurate order data. This will allow the order to be processed in x days and inventory to be updated within y hours.

2.2 Medium Priority

1. Automated testing
2. Terms and Conditions, Disclaimers, and Other Legal Jargon
3. Contact us Page

2.3 Low Priority

1. The action triggered mail/SMS
2. Cellular device authentication
3. Multi-language support
4. Filter and Sort functionality in searching
5. Product review and Relevant suggestions
6. Chatbot support

3. Technical Objectives

For the purpose of this app, we have chosen the Tech stack that we thought would be the best one, enabling us to speed up the development process, Cut down on testing time and bugs. We Have had these basic requirements to keep in mind while selecting the Tech Stack

The main Technology used here is React-JS for the front-end with a NodeJS + google firebase backend. This decision was taken keeping in mind the hosting, Database management needs for the app, and secure, easy, and trusted Authentication protocols that have been in the trade for a long and have been tested.

We will implement the Stripe/Razorpay functionality with either the cloud functions (nodeJS) or the Google Firebase protocols

3.1 Reliability

- LTG shall be completely operational and fast at all times

To keep the App reliable and up and running at all times, Team Delta will Host the app on Netilfy, which will enable the user to hit the web app instantaneously with a minimal amount of wait time.

3.2 Usability

- A user should be instantaneously able to see the profile details
- A user who already knows what product he is interested in should be able to locate and view that page

Keeping the usability factor in find, we are going with the easy, minimal design with a focus on the product rather than making the site bulky with animations. The search functionality will be implemented by a ReactJs library “ react-fuzzy” and “fuseJS” which will enable the developers to implement fuzzy search functionality.

The react fuzzy and fuseJS are easily compatible with either the REDUX store or the Context Api that Team Delta is choosing for content-prop management on the client-side. The reason for using the content management system in the app is solely performance and implementing a minimum bug code. Prop drilling can introduce hard-to-fix bugs and can essentially make the user experience very tedious.

3.3 Performance

- The system should be able to support multiple simultaneous users

Keeping the performance aspect in mind the Team Delta will be using the ReactJS coupled with CSS (Bootstrap, SaSS) on the client.

And for the backend, NodeJS + MongoDB coupled with Google's Firebase is used. The decision of using Google Firebase was taken with its real-time database (firestore) which will allow the developers to store the inventory and display it instantaneously onto the client-side.

The cloud functions for the Stripe payment gateway will be implemented with Node JS cloud functions allowing the developers to speed up the development process.

3.4 Security

- The system shall provide password-protected access to web pages that are to be viewed only by employees.
- Transaction data must be transmitted in encrypted form.

To make the app secure, a Google Authentication coupled with a NodeJS authentication will be implemented, wherein the user can log in / register into the correct account, with speed and security.

Firebase provides an easy implementation of the authentication and thereby making the app more secure, and have a faster implementation process

Users will also be able to log-In via their cellular device.

3.5 Supportability

- The system should be able to accommodate new products and product lines without major reengineering.
- The system website shall be viewable from Internet Explorer 4.0 or later, Netscape Navigator/Communicator 3.0 or later, and the America Online web browser version 3.0 or later, Google Chrome, Mozilla Firefox, Microsoft edge (all previous and current versions).

3.6 Interfaces

The system must interface with

- The current NoSQL database (MongoDB and Firestore)for product and order information
- The current Stripe/Razorpay Payment Gateway

3.6 Databases and Data Storage

The system will have two data storages :

- A database to store the User Profile details (Name, addresses, Payment Details.) will be done via the Firebases' database FIRESTORE.
- A second database to store the actual inventory with the prices, description, and other details.

The databases used in the LTG are NoSQL databases, This decision was taken due to the following reasons:

1. Scalability :
    - NoSQL databases are horizontally scalable. This means that more traffic can be handled by sharding, or adding more servers to the NoSQL database

2. Schema Type:
    - NoSQL databases have a dynamic schema type and hence are better suited for big data, which makes them more flexible

Also, we'd like the app to be a low latent one, which makes the decision of using a NoSQL database (MongoDB and Firestore) a better option for us rather than using a SQL database

MongoDB is our choice of database for inventory data storage due to following reasons:

Why MongoDB?

- Document Oriented Storage − Data is stored in the form of JSON-style documents.
- Index on any attribute
- Replication and high availability
- Auto-Sharding
- Rich queries
- Fast in-place updates

Why Firestore?

- Cloud/Serverless solution
- Offers Atomicity, Consistency, Isolation, and Durability (ACID)
- Less data Flattening or denormalization
- Scalable
- Robustly Secure
- Pay-as you go Model thereby minimizing development cost

Moreover, integration of the Stripe payment gateway is very secure and also we'll easily able to integrate the SSL encryption for the data storage for the delicate user information like the Payment cards.

The Data Tables

a) User :

User Id	Integer
Name	String
Email	String
Address	String
Contact Info (phone Number)	String

- The team delta won’t be saving the Payment details of the customer as there is simply far too much risk involved, and you will typically need to be externally audited to ensure that you're complying with all the relevant local laws and security practices.

b) Inventory :

ItemId	Integer
Item Category	String
Item Name	String
Description	String
Price	Number
Item image	String (URL)

5. The Use Case Model

5.1 Use Case Descriptions (for selected cases)

Notes:

- For all use cases, the user can cancel the use case at any step that requires user input. This action ends the use case. Any data collected during that use case is lost.
- For all use cases that require a logged-in user, the current login session is updated during the use case to reflect the navigation paths through the use case.

Login User

Use Case Name:	Login User
Summary:	To get personalized or restricted information, place orders, or do other specialized transactions a user must log in so that the system can determine his access level.
Basic Flow:	1. The use case starts when a user indicates that he wants to log in. 2. The system requests the username and password. 3. The user enters his username and password. 4. The system verifies the username and password against all registered users. 5. The system starts a login session and displays a welcome message based on the user's preferences.

Alternative Flows:	Step 4: if a username is invalid, the use case goes back to step 2. Step 4: if the password is invalid the system requests that the user re-enter the password. When the user enters another password the use case continues with step 4 using the original username and new password.
Extension Points:	none
Preconditions:	The user is registered.
Postconditions:	The user can now obtain data and perform functions according to his registered access level.
Business Rules:	Some data and functions are restricted to certain types of users or users with a particular access level.

Register User

Use Case Name:	Register User
Summary:	To get personalized or restricted information, place orders, or do other specialized transactions a new user must register a username and password.

Basic Flow:	1. The use case starts when a user indicates that he wants to register. 2. The system requests a username and password. 3. The user enters a username and password. 4. The system checks that the username does not duplicate any existing registered usernames. 5. The system requests a name (*), street, city, state, pin code(*), phone, and email address. Items marked by (*) are required. 6. The user enters the information. 7. The system determines the user's location and access level and stores all user information. 8. The system executes use case *Register Preferences.* 9. The system starts a login session and displays a welcome message based on the user's preferences.
Alternative Flows:	● Step 4: If the username duplicates an existing username the system displays a message and the use case goes back to step 2. ● Step 5: If the user does not enter a required field, a message is displayed and the use case repeats step 4.
Extension Points:	*Register Preferences*
Preconditions:	none
Postconditions:	The user can now obtain data and perform functions according to his registered access level.
Business Rules:	● A registered user's location is the LTG location nearest his zip code. ● Access levels are ○ 0: A user can access only data classification 0 ○ 1: The user can access data classification <= 1 ○ 2: The user can access data classification <= 2 ● The default access level is 0.

Register Preferences

Use Case Name:	Register Preferences
Summary:	This use case allows a registered user to enter or change his preferences.
Basic Flow:	1. The use case starts when a user indicates that he wants to enter or modify his preferences. 2. The system displays all current product lines. It indicates any product lines that the user has currently selected. 3. The user selects/deselects product lines. 4. The system displays current language preferences. It indicates the language preference currently selected. 5. The user may select a different language preference. 6. The system stores any change to language preference.
Alternative Flows:	none
Extension Points:	none
Preconditions:	The user is logged in.
Postconditions:	The system can customize a welcome message based on the user's revised preferences.
Business Rules:	Language selections allowed is English (default)

Place Order (Customer)

Use Case Name:	Place Order Scenario: Customer places his order.
Summary:	This use case allows a registered customer to place an order for a

	product.
Basic Flow:	1. The use case starts when a customer indicates he wants to place an order for the current product being displayed. 2. The system displays the customer's information: name, street, city, zip, phone, email. 3. The customer may add or change any of the information. 4. The system stores any changes. If the zip code has changed, the system modifies the customer's location. 5. The system requests the quantity to order and the shipping address. 6. The customer enters quantity and shipping address. 7. The system displays the payment options available to this customer. 8. The customer selects a payment option. 9. The system completes the payment by executing the use case *Charge Customer* 10. The system stores the order information decreases the quantity on hand for the product and sends the order details to Shipping. 11. The system displays an order completion message and sends a receipt to the user.
Alternative Flows:	Step 9: If the selected payment method could not be validated, go to step 8 to get another payment option. Step 10: If the quantity on hand is not sufficient for this order, a message is sent to the customer and the use case is canceled.
Extension Points:	*Charge Customer*
Preconditions:	The customer is logged in and has completed a search for the product to be ordered
Postconditions:	The product is sold.

Business Rules:	If a customer has been previously authorized for billing by a sales agent, the customer may be billed for the order. Otherwise, the customer must pay in full by credit card at the time of the order.

Charge Customer

Use Case Name:	Charge Customer
Summary:	This use case charges the order currently being placed to a credit card.
Basic Flow:	1. The use case begins when a user selects "Credit Card" as a payment option, while in use case *Place Order* 2. LTG requests the credit card number, type, and expiration date. 3. The user enters the information. 4. LTG verifies that the credit card is valid for the amount to be charged and completes the credit card transaction. 5. The system stores the payment details and returns a success message
Alternative Flows:	Step 4: If the credit card cannot be validated the use case ends, returning a failure message
Extension Points:	none
Preconditions:	LTG is executing the use case *Place Order*.
Postconditions:	The customer has been charged for the order.
Business Rules:	Credit cards accepted are Visa, MasterCard, and Discover.