Inventory Management App Guide

Table of Contents

[Introduction to the Inventory Management MEAN Application 1](#_Toc173665896)

[Login and Registration Process Walkthrough 1](#_Toc173665897)

[Dashboard Overview 5](#_Toc173665898)

[Notification Overview 8](#_Toc173665899)

[Logout Overview 12](#_Toc173665900)

## Introduction to the Inventory Management MEAN Application

This application, built using the MEAN stack (MongoDB, Express.js, Angular, and Node.js), leverages the strengths of each technology to provide a seamless and efficient user experience.

**Key features:**

* **User Authentication:** Secure login and registration functionalities ensure that only registered users can access and manage inventory data.
* **Real-Time Inventory Updates:** The application supports real-time updates, allowing users to see changes instantly without needing to refresh the page.
* **Notification System:** Users can enable or disable notifications for various actions, such as adding, updating, or deleting inventory items, to stay informed about the latest changes.
* **Inventory Management:** Easily add, update, and delete inventory.

This document will provide a comprehensive tour of the application, covering each view offered by the app, its features, and the underlying database structure. By the end of this guide, you will have a thorough understanding of how to navigate and utilize the Inventory Management Application to its full potential.

## Login and Registration Process Walkthrough

The Inventory Management App begins with a streamlined user authentication process, providing users with a clean and intuitive interface for both logging in and registering a new account.

**Login Screen**

A screenshot of a login screen

Description automatically generated

The login screen, accessible at <http://localhost:4200/login>, presents users with fields for their email and password. The form utilizes Angular’s reactive forms for validation, ensuring that users provide the necessary information before proceeding.

* **Email Field:** This field is required and must contain a valid email format. If the user fails to provide a valid email, an error message will prompt them to correct their input.
* **Password Field:** Users must enter a password that is at least 8 characters long. Any attempt to submit the form without meeting this requirement will trigger an error message.

After entering the required information, users can log in by clicking the “Login” button. The form submission is handled through Angular’s reactive form bindings, which ensures that all validation is in place before any data is sent to the backend for authentication.

**Registration Screen**

A screenshot of a login form

Description automatically generated

For new users, the registration screen is available at <http://localhost:4200/register>. The registration process closely mirrors the login process with the addition of backend checks to ensure that the provided email is not already in use.

* **Email Field:** As with the login form, this field requires a valid email format. Users will be notified if the email they provide does not meet the required criteria.
* **Password Field:** Passwords must be at least 8 characters long. Users will be notified if their password does not meet this criterion.

Once the user fills out the form with a valid email and password, they can create an account by clicking the “Register” button. The form will then send the data to the backend, where it is checked for duplicates and securely stored if the email is unique.

Upon acceptance of the registration, the user is presented with a Toast message indicating the registration was successful and the user will be redirected back to the login screen to enter in their login credentials.

**Populating the Users Table Upon Registration**

A screenshot of a computer

Description automatically generated

The screenshot above shows the **users** collection within the MongoDB database, specifically within the **inventoryapp** database. This collection stores information related to registered users of the Inventory Management app.

Each document in this collection represents a registered user and includes the following key fields:

* **\_id:** This is the unique identifier automatically generated by MongoDB for each user.
* **Email:** The email address used by the user during registration.
* **Password:** A hashed version of the user’s password, ensuring that sensitive information is securely stored.
* **notificationsEnabled:** A boolean value indicating whether the user has enabled notifications (enabled by default).

This collection is essential for managing user accounts within the application, ensuring that each user has a unique identity and secure credentials. The data in this table directly correlates with the registration process, where new users are added to this collection upon successful registration.

## Dashboard Overview

**Viewing Inventory**

**A screenshot of a computer

Description automatically generated**

The dashboard view, accessible at <http://localhost:4200/dashboard>, displays a table of existing inventory items. Each row represents an item with columns for the item name, SKU, quantity, and actions. The actions column includes buttons for editing and deleting items.

**Adding a product**

A screenshot of a computer

Description automatically generated

To add a new product, click the “Add Product” button, which opens a dialog window. This window includes fields for the product name, SKU, and quantity. Once the required fields are filled out, clicking “Create” will save the new product to the database.

**Editing a Product**

**A screenshot of a computer

Description automatically generated**

Clicking the “Edit” button next to an inventory item allows you to modify its details. Editable fields include the item name, SKU, and quantity. After making changes, clicking “Save” will update the product information in the database.

**Deleting a product**

The “Delete” button allows you to remove an item from the inventory. When clicked, the item is deleted from the database, and the table updates to reflect this change.

**Notifications and Confirmation**

For each action (adding, editing, deleting), a Toast notification provides confirmation. Additionally, if notifications are enabled, a new notification is added to the notification list on the notification page (covered later in this guide).

**Inventory Items Collection Overview**

**A screenshot of a computer

Description automatically generated**

The screenshot above displays the **inventoryitems** collection within the MongoDB database for the Inventory Management Application. This collection stores the documents representing individual inventory items, with each document containing fields such as **name, sku, and quantity**. The **\_id** field is automatically generated by MongoDB and serves as the unique identifier for each item.

In this specific view, you can see three inventory items, each with a unique **\_id**, a ­**name** representing the item name, an **sku** to track inventory, and a **quantity** representing the available stock of that item. This collection is central to managing the inventory within the application, with CRUD operations performed through the dashboard, directly impacting the data stored here.

## Notification Overview

A screenshot of a computer

Description automatically generated

The **notification view**, as shown in the screenshot above and accessible at <http://localhost:4200/notifications>, serves as a centralized location where users can view, manage, and delete notifications related to inventory activities. Each notification is displayed with a message describing the event (e.g., adding, updating, or deleting inventory) along with a timestamp. The user can delete any notification by clicking on the trash icon next to it, which will remove it from both the view and the database.

The **Enable Notifications** toggle at the top of the view allows users to control whether notifications should be generated and stored for their account. If notifications are disabled, new notifications will not be generated following inventory management events.

**Notifications Collection Overview**

**A screenshot of a computer

Description automatically generated**

The notifications collection, as displayed in the MongoDB screenshot above, is where all notifications are stored within the application. Each document in this collection represents a single notification and includes fields such as **\_id, msg, and date**. The **msg** field contains the message displayed to the user in the notification view, while the **date** field records the exact time when the notification was created. Each notification is stored in this collection and can be retrieved and displayed in the notification view until it is deleted by the user.

**Explanation of the Notification Status Process**

When a user toggles the notification setting on the Notifications page of the Inventory Management App, several actions take place to ensure the user’s preference is stored and respected across sessions.

**Local Storage**

A screenshot of a computer

Description automatically generated

When the user enables notifications, their **userId** and **notificationsEnabled** status are stored in the browser’s local storage, allowing the application to quickly check and apply the user’s preferences on load. The image above displays the cookie values recorded when notifications are **enabled**. The image below shows the updated **notificationsEnabled** value when **disabled**.

A screenshot of a computer

Description automatically generated

**Database Update**

A screenshot of a computer

Description automatically generated

The application doesn’t just store the notification preference locally but also ensures this preference is saved in the database for persistence. After the user disables notifications, the application updates the corresponding user document in the MongoDB database, setting the **notificationsEnabled** field to **false**, as displayed in the image above. This ensures that the user’s preference is maintained even if they log in from a different device or browser.

This dual storage method (in both local storage and the database) guarantees that the user’s notification preferences are respected consistently across difference sessions and devices.

## Logout Overview

A user can logout by simply clicking the logout link in the navigation bar. This will end the user’s session and redirect the user back to the login screen.