Tampere University of Applied Sciences

Developing an Inventory Management Application Using MERN Stack

Md Iftekher Rasel

Raonaq Lubna

BACHELOR'S THESIS

November 2024

Degree Programme in Software Engineering

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Bachelor's Degree Programme in Software Engineering

RASEL, MD IFTEKHER & LUBNA, RAONAQ:
Developing an Inventory Management Application Using MERN Stack
Bachelor's thesis 86 pages, appendices 1 page
November 2024

Efficient inventory management is vital as businesses grow, yet many in Asia still rely on outdated manual methods like spreadsheets, which are error-prone and lack scalability. Modern inventory management systems offer centralized, real-time solutions to streamline operations across multiple branches.

The Purpose of this thesis project was to modernise and streamline the manual processes by developing an updated inventory management system that addresses the limitations, providing real-time tracking, multi-branch support, and automation.

The MERN stack (MongoDB, Express, React, Node.js) was chosen for building the proposed solution and by using it, organizations can streamline their inventory processes and mitigate operational challenges associated with inventory control and resource management. Small start-ups seeking scalable inventory management solutions, mid-sized enterprises aiming to optimize resource allocation, and large corporations striving for enhanced operational efficiency can all find value in the outcomes of this research.

In this project, Rasel handled the backend development using Node.js, while Lubna focused on the frontend development with React.js. Backend testing was performed using Postman to ensure the server-side functionality was working as expected. The MongoDB database was used locally for data storage and testing. After completing the backend, the server was uploaded to GitHub, from where Lubna pulled the changes to work on the frontend. Once the frontend was updated, changes were pushed back to GitHub for further integration.

Key Words: MERN stack, MongoDB, Express, React, Node.js

CONTENTS

ABBREVIATIONS AND TERMS

AI	Artificial Intelligence
API	Application Programming Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	Hypertext Markup Language
IMS	Inventory Management System
KPI	Key Performance Indicator
MERN	MongoDB, Express.js, React.js, and Node.js
MVC	Model, View, and Controller
ODM	Object Data Modelling
RAD	Rapid Application Development
RFID	Radio-Frequency Identification
SME	Small and Medium Enterprise
SQL	Structured Query Language
UPC	Universal Product Code
UI	User Interface
URL	Uniform Resource Locator

1 INTRODUCTION

In the modern software world, effective inventory management is a key component for many businesses, manufacturing company and retailer. As the firms are starting to surge in complication along with size, trustworthy and flexible inventory management solutions are in high demand. Considering the fact, MERN stack might be the best possible technology that offers combination of adaptability, efficiency, and developer friendliness. MERN stack comprises MongoDB, React.js, Node.js and Express.js. The main objective of this thesis is to develop MERN stack as a base framework for the inventory management system (IMS) which gives organizations to solve various difficulties that is related to inventory management systems. The project uses MongoDB for flexible data storage, React.js for dynamic user interfaces (UIs), Express.js for server development, and lastly Node.js for backend operations.

Moreover, the efficiency of any software solution is tied to its ability to function within a wide ecosystem. Thus, comprehensive testing emerges as an indispensable facet of software development, ensuring the reliability, and performance security of applications. In the project, Postman works as a preeminent tool for API testing helping to validate endpoints, system interoperability, and lastly data integrity.

Implemented thesis not only uses API testing through Postman in fortifying the functionality of inventory management systems but also explores the inherent capabilities of the MERN stack. By crafting the prowess of MERN with the Postman precision, organizations can authorize themselves with a scalable and resilient inventory management solution in the age of modern business environments. The main objective of this thesis is to illuminate the potentiality of MERN stack by rigorous API testing, maintaining efficiency and competitiveness in the inventory management practices.

1.1 Current Challenges with Excel Sheets

If we study the history, small businesses usually use Excel sheets for monitoring and managing their variety of aspects product inventory. Nonetheless, this approach is followed with a lot of difficulties and time consumption as well as a high risk of human error. Some representative difficulties and problems are presented below.

1) Scalability Issues: Excel documents are characteristically bounded in their scalability, which leads to a difficulty with expanding inventories.

2) Error-proneness: Conventional process of data entry in Excel documents is susceptible to human error, which can drag to imprecisions in inventory records followed by affecting business operations.

3) Absence of Real-Time Insights: One of the notable drawbacks of Excel-based systems is inability to synchronize real-time updates. Thus, the ability to make timely decisions is hampered.

4) Inefficiency of Communication: Version complexity can be seen, which may lead to an inefficiency of Excel files sharing among team members.

1.2 Complexity of Inventory Management in Large and Small Companies

At the beginning of startup or small business, Excell sheet seems to be advantageous as well as user friendly, but as the business grows, they soon realize the limitations of such tools. Excel sheet shows lack scalability and cannot effectively manage inventory activities which leads to drawbacks such as inaccuracies, inefficiencies, and enlarged risk of errors. Consequently, small companies may find themselves in a complexly manual process, encumbering their ability to meet market demands and possible growth opportunities.

Contrariwise, large companies often seem to deal with a diverse set of challenges in inventory management. Due to their sheer scale and complexity in operations,

Excel-based systems is completely disadvantageous and impractical to them. Managing their massive inventory quantities, suppliers and distribution channels entails an erudite solution which is capable of handling diverse data sets followed by uninterrupted communication between stakeholders. For this specified reason, Excel sheets falls short of its performance, accuracy, decision making speed and so on.

1.3 Impact of the Project

Implemented project helps to shift in a new management practice by leveraging the MERN stack that helps to tailor a business solution of all sizes. The main objective of this project is to enhance the whole inventory management process, digitalize the system and finally empower organizations to overcome the limitations of traditional Excel-based systems. Using of MongoDB as a flexible scalable database, Express.js for building backend APIs, React.js for creating user interfaces, and lastly Node.js for server-side scripting helps to imperfect this project. Overall, MERN stack serves as a strong foundation for developing a solid inventory management system.

APIs mainly ensure the reliability, security and performance of the inventory management system whereas Postman helps to enhance the overall functionality of the software.

The impact of this project lies in its ability to initialize the advanced inventory management capabilities and making them reachable to all sizes of organizations. Transforming from conventional Excel-based system to modern and digitalized solution powered by the MERN stack and continuous API testing, any businesses can improve their efficiency, accuracy and agility in their inventory practices. Whether a start-up company or growing small company looking to progress or a multinational organization looking to heighten its operations, the proposed project offers an effective and transformative approach to inventory management which will drives sustainable growth of the organization and a competitive advantage in the current business market.

2 LITERATURE REVIEW

Inventory management started long ago when merchants and traders sought a mechanism to keep track of their goods. Better and more intelligent methods of inventory management were discovered over time. Initially, the majority of techniques were carried out by hand, with employees physically noting stock levels. Better systems were required as businesses became more complex and larger. In order to determine how much inventory management has advanced and where it is headed, let's examine some of its pivotal milestones.

2.1 Manual Counting

Prior to the Industrial Revolution, inventory management was extremely straightforward. Shopkeepers and businessmen were among the first to use it. They needed to count their products at the end of each day to see how many were sold and plan for future needs. This counting was done by hand and typically took several hours or even days. It was also not always exact because mistakes were common, such as missing goods or incorrect counting. This strategy helped businesses comprehend their stock and make future plans, despite the fact that it was straightforward and time-consuming. It established the groundwork for the sophisticated inventory management systems we currently employ.

2.2 Machine Readable Punch Card

Machine-readable punch cards were one of the earliest steps toward employing technology to control inventory. In 1889, Herman Hollerith created the first punch card, which used tiny holes in paper to gather information for things like work hours and censuses. This served as inspiration for Harvard's check-out system, which was developed in the 1930s and used punch cards for inventory and billing. After being read by a computer, the cards that customers filled out were delivered

to the warehouse for item delivery. Similar systems are still in use for pricey or restricted commodities, such as medications, but they weren't very popular because they were slow and expensive [17].

2.3 The Modern Barcode

It was in the late 1940s and early 1950s that the first barcode was created. Special ink that responded to ultraviolet light was used, along with a reader. The system was too large and ineffective, though, because the technology wasn't sufficiently developed. In the 1960s, a group of retailers developed the modern barcode for inventory tracking. Scanners become faster and more affordable thanks to laser technology. The Universal Product Code (UPC) was the first widely used barcode, introduced in the late 1960s. On June 26, 1974, in a Marsh grocery store in Troy, Ohio, the first barcode was scanned on a pack of Juicy Fruit gum. [17].

2.4 Software Improves Tracking

Inventory tracking improved dramatically in the 1980s and 1990s thanks to advances in computers and software. These systems operated in a loop, tracking inventory, monitoring it, making purchases, and starting over. As personal computers grew increasingly popular, barcodes and scanners got considerably more affordable. However, because they lacked a location to keep all of the data they gathered, many small and medium-sized firms were slow to implement barcode systems. Inventory was no longer being manually tracked by this point. Rather, they scanned objects and input the information into computers. Software for inventory management got significantly better in the early 2000s. Barcode scanners allowed businesses to automatically update their databases, eliminating the need for manual data entry. [17].

2.5 The Introduction of RFID in Barcode Technology

Since its initial patent in the 1970s, radio-frequency identification (RFID) technology has been widely used in factories, warehouses, and retail establishments. Product data, like the product kind, manufacturer, and serial number, are sent to a scanner via a microchip. Considered as a more sophisticated barcode. RFID tags are perfect for reaching high shelves in warehouses since scanners can read them even when they are not in direct line of sight. Additionally, RFID tags have a larger data storage capacity than a standard barcode [17].

2.6 Earlier Research

A paper by Guido van Heck published in 2009 as a thesis work in Delft University of Technology. The creation of an inventory management performance measurement tool was the goal of this study. Consequently, a distinct framework was created that allowed organizations to assess the effectiveness of their operational inventory management. The framework has five business process steps, with meaningful KPIs associated to each step for measurement. A certain collection of measurements is provided by the framework, which is organized along the entire process from start to finish. This makes it possible for users to observe how the many facets of inventory management are cohesive. Because a predetermined set of measures is included in the framework, inventory management performance can also be measured in an organized manner. Now the framework sounds like a sentient entity of evaluating the inventory management performance. In 2023, a paper was published by Chan Chin Wei, Sathiapriya A/P Ramia and Nurul Farhaini Razali in Journal of Applied Technology and Innovation. The goal of this research was to create an Inventory Management System (IMS) that can improve management and handling of product stock, customer orders, customer service, and order delivery in relation to corporate inventory information. The target user is the owner and employee of a Small and Medium Enterprise (SME) retail store that currently maintains inventory manually. IMS enables retail stores to track the upcoming arrival of product stocks and record consumer orders for product reservations in the store inventory. The developer in this study developed frontend systems using HTML, CSS, and JavaScript, and

backend systems using PHP. The software methodology known as Rapid Application Development (RAD), which places a strong emphasis on an iterative development process, is also used in this study [13].

2.7 Related Research / Published Works

In 2022, Rishabh gupta, Ashish and Aman Yadav published a paper in International Journal of Creative Research Thoughts. In this research work, MERN stack technologies were used. The resulting straightforward desktop program allows information to be updated and verified within the store by connecting to the specific distribution centre. Additionally, daily, weekly, and monthly sales data is provided by it. Benefits of the inventory management system include higher revenue, improved profitability, and general improvements in customer satisfaction [8].

In 2017, Dr. M. Sujithra , Hanish S , Akschaya B , Danvanth S , Sivasakthi G published a paper on International Journal of Research Publication and Reviews. Building a product catalogue application using the MERN stack is explained in detail in this paper. The program was made to save product characteristics, including pictures, descriptions, and classifications, and then present them in an intuitive way on the front end. Additionally, the program offers option for adding and removing products, which modifies the database in line with those changes. This paper presents the study's findings and talks about the technology and technique utilized to construct the application [9].

In 2023, Prof. Yogesh Kadam, Akhil Goplani, Shubit Mattoo, Shashank Kumar Gupta, Darshan Amrutkar, and Prof. Dr. Jyoti Dhanke published a paper on European Chemical Bulletin. The paper discussed about the benefits of the MERN stack over earlier technologies like HTML, CSS, SQL, and NoSQL along with the reasons why it is so popular. A succinct summary of the MERN stack's elements, features, and use in web application development is also included in the article. The MERN stack has seen a rise in popularity in web development during the past few years. Every element of the MERN stack serves a distinct purpose, and when combined, they offer a full-stack development environment that makes it possible for programmers to efficiently design web applications [10].

3 METHODOLOGY, MODELING AND PROJECT IMPLEMENTATION

Scientific revolution on inventory has taken place over the planet. As a result, the necessity for automatic and advanced inventory management system is increasing from day to day. MERN stack for inventory management might be a valuable resource in fulfilling this requirement. The technique and modelling of the project are discussed and described in this chapter. This chapter involves the design, building and software specification used for the proposed project.

3.1 Flow Chart

Figure 1 shows the flowchart of the proposed project. The user needs to launch the webpage first. They have to visit the login page if they already have an account. If the user isn't registered, then needs to sign up. After signing up, user needs to wait for the admin approval. The user will be able to log in if the administrator grants permission. The user will have to reapply if the administrator rejects the account. After log in, user will have authorization and the system will generate an authorization token. Under this part there will be three sub trees:

- If the user is branch admin, then system will take user to the branch admin dashboard.
- If the user is employee, then it will go to employee dashboard.
- If the user is head admin, then it will go to head admin dashboard.

After reaching out to the head admin dashboard, the "Head Admin Operation" took places. In that corresponding section, four main works can be controlled:

a. User management:
    - Update user info
    - Remove user
b. Location management:
    - Add a location
    - Remove a location
c. Product management:
    - Add product / approve product
    - Remove product

d. Inventory management:
    - Approve the pending inventory
    - Remove pending inventory.

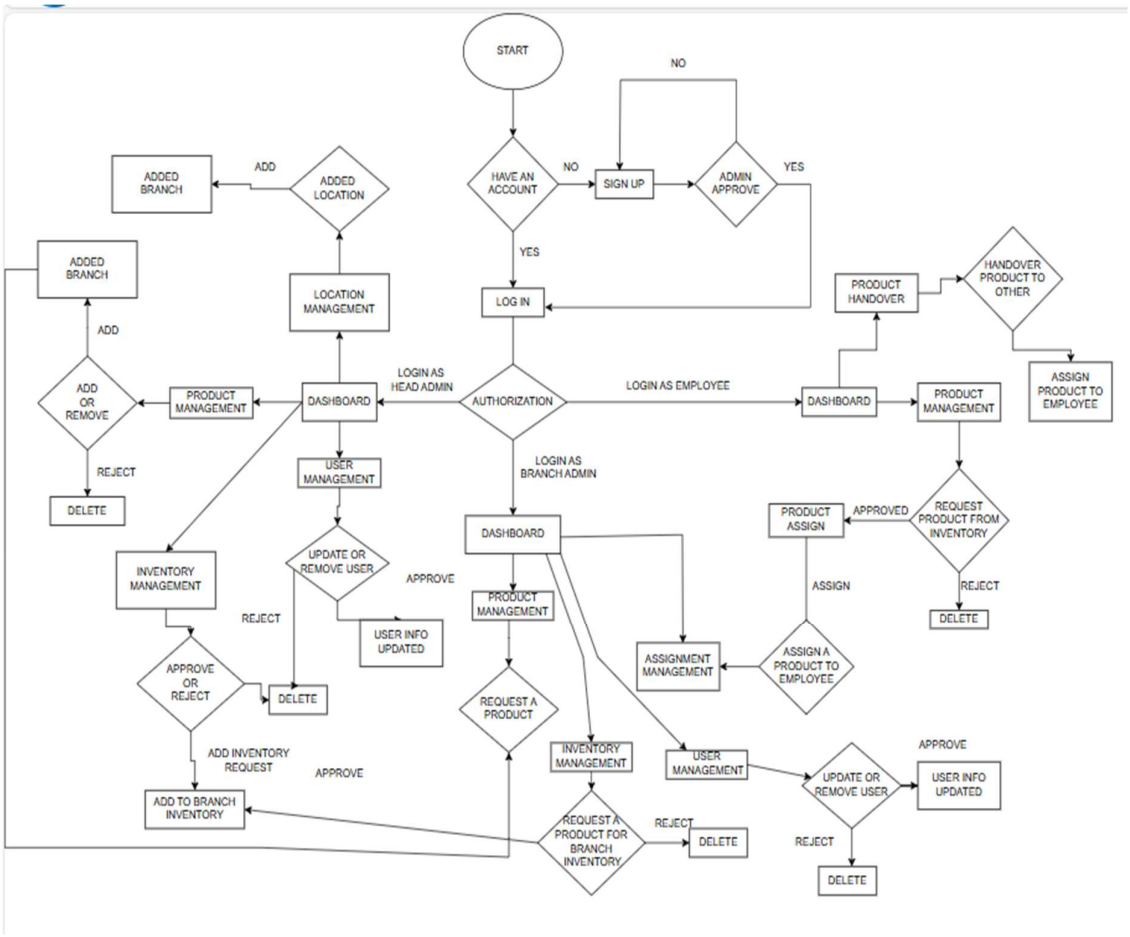


Figure 1. Flowchart of the project.

After the completion of head admin operation activity, the “Branch Admin Operation” is followed by. In this section again four operations will be checked:

a. User management:
    - Approve a user
    - Update a user
b. Product Management:
    - Request a product for adding in the central inventory
c. Assignment management:
    - Assign a product to other employees
    - Approve product for employee request
d. Inventory management:

- Add product to the branch inventory
- Update product in the branch inventory.

Finally, comes the “Employee dashboard” section. Under this section, two objectives will be monitored:

a. Product management:
    - Add a product request to the branch inventory
    - Update product request
    - Show the assigned product
b. Handover management:
    - Transfer product to the other employee.

3.2 Tools or Software Required

To complete this project, one valuable tool that has been utilized is Postman. In this part, a detailed discussion has been given about used software tools or technology.

3.3 MERN Stack

Based on JavaScript technology, MERN stack is a powerful and extremely productive framework for front-to-back development of single-page applications (SPAs). With its comprehensive workflow and integrated technology, MERN offers developers a tech-driven methodology that expedites development. The MERN stack is now widely used for web app development across numerous industries. These consist of social networks, productivity tools, and news aggregation. Frontend and backend technology are areas of complete expertise for MERN stack engineers. It's critical to create dynamic, multipurpose web platforms. Within the MERN stack, developers modify workflows to accommodate changing requirements, improve frontend and backend capabilities, and integrate APIs [11]. Full stack developers can learn a great deal about a variety of related tools and technologies by using MERN. These include webpack, Git, HTML, CSS, JavaScript, TypeScript, and a few testing frameworks. Backend APIs which is

developed by Node.js from MERN stack framework can also collaborate with others to create a visually appealing and scalable online application. There are four primary parts to the MERN stack: Node.js, React.js, Express.js, and MongoDB. Every element in the stack has a distinct function during the development phase [6]. Figure 2 shows a basic architecture of MERN stack development.

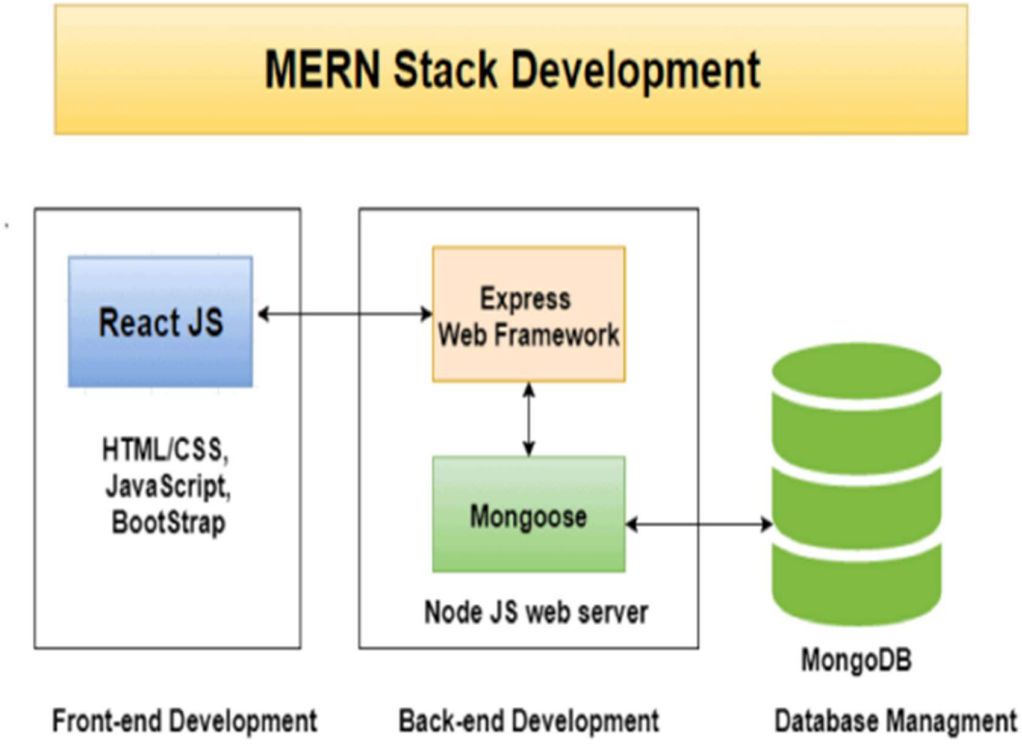


Figure 2. MERN stack architecture [15]

3.4 MongoDB

The MERN stack's data storage component is a NoSQL database called MongoDB. MongoDB, which is well-known for its scalability and versatility, is an excellent option for managing massive volumes of data. MongoDB facilitates seamless integration with JavaScript-based apps by storing data in documents that resemble JSON. It is an effective tool for maintaining and retrieving data in MERN applications because of its query language and data manipulation features [12].

3.5 Express.js

A lightweight and straightforward framework for creating web apps and APIs is called Express.js. It is based on Node.js and facilitates the use of middleware, routes, and HTTP requests Express.js allows developers to quickly and effectively build robust server-side applications. It facilitates seamless front-end and back-end communication by integrating nicely with other components of the MERN stack [12].

3.6 React.js

The front-end component of the MERN stack, React.js is frequently used when creating UIs. React is based on a component-based architecture. This allows UI component developers to create and manage reusable components. React.js offers a virtual document object model (DOM) that effectively updates only the UI elements that are required, optimizing rendering efficiency. React is an effective tool for developing dynamic and interactive web apps because of its declarative syntax and effective rendering [16].

3.7 Node.js

One popular JavaScript environment is Node.js. In particular, a runtime environment that lets programmers use JavaScript code outside a web browser. It functions as the MERN stack's backend element. Node.js an extremely scalable and effective event-driven, non-blocking I/O approach. It enables code sharing between the front end and back end by enabling developers to create server-side apps using JavaScript. Because of its extensive ecosystem of modules and packages, Node.js is simple to integrate with other programs and technologies [16].

3.8 MVC Architecture

The MVC architecture is made up of three major components: the Model, View, and Controller. Trygve Reenskaug created MVC in 1979 to offer superior solutions for big, complicated problems. It was first implemented in the Smalltalk-80 framework, which was used to create Apple interfaces (Macintosh and Lisa) (Reenskaug, 1979). The MVC pattern is well known for its advantages in the construction of web applications, but it may also be used in the building of desktop applications.. The business layer (Model logic), the presentation layer (View logic), and the control layer (Controller logic) are divided into distinct layers in MVC architecture. In order to reduce complexity and work on a single file MVC helps organize the data and split the code according to requirements. The MVC design can be used to gain code flexibility and reusability. Figure 3 illustrates the flow structure of MVC.

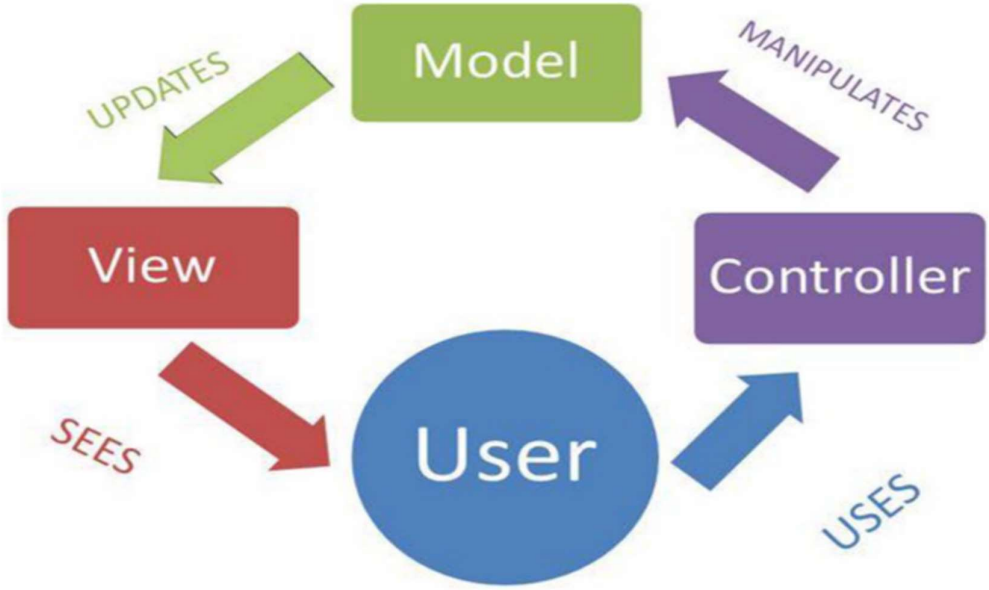


Figure 3. MVC Architecture [3]

3.8.1 Model

The Model is an essential part of an MVC application that controls the data and business logic. It arranges the information and gives it to the View, which is in charge of presenting the result to the user. It oversees the data for the application. In essence, it includes the application data, function specification and formulation of logic. A model could be an individual object or a combination of objects. In addition to managing data, this layer permits database communication, which includes inserting, retrieving, and updating data in the database [14].

3.8.2 View

Views is used to design the program's user interface. The user interface is used by users to interact with the web pages. View displays the outcomes of the data included in the model. The View is responsible for showing Model information. The superfluous properties are hidden and just the necessary attributes are displayed. As a result, it offers us the benefit of presentation summarization. View can be readily connected with AJAX technologies, just like JSP, ASP, and PHP tags [4].

3.8.3 Controller

In order to reply to user queries, the Controller layer is utilized. Controller carries out the user's requests. The controller serves as a go-between for the user and the system. Controller is in charge of both View and Model. It manages the flow of data within the model and instantly refreshes the view when there is a change in the data. The controller acts as the connection point between the model and the view. It processes user inputs from the View and updates the Model as necessary. It basically aids in controlling the relationship between the data and reasoning that underlie what the user sees (the View) and the data itself (the Model). [4].

3.9 Mongoose

A Node.js utility called Mongoose facilitates interaction with MongoDB. It links code to MongoDB data, arranges data, preserves data relationships, and makes ensuring data complies with rules (schema validation). MongoDB is a NoSQL database that stores data in the form of flexible JSON documents. Unlike SQL databases, it does not require a predefined structure, making application development faster and deployment simpler [1]. A connection between Node.js and MongoDB with Mongoose is shown in Figure 4.

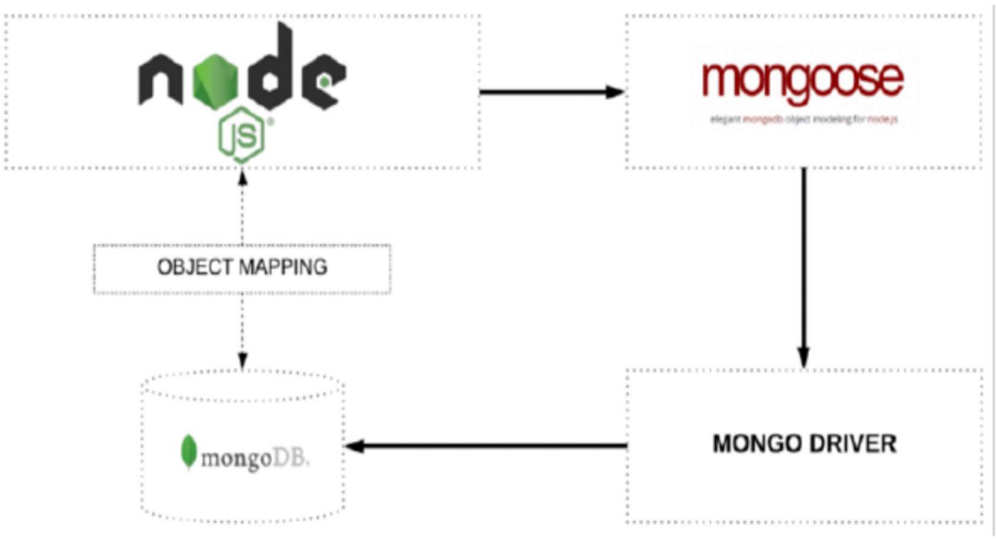


Figure 4. Object mapping between Node.js and MongoDB managed via Mongoose [5]

3.10 Bootstrap

The ‘*’Bootstrap in development”* usually refers to the use of the well-known front-end framework during the web application or website development process. With the use of pre-designed HTML, CSS, and JavaScript components and styles from Bootstrap, developers can produce user interfaces that are both visually appealing and responsive. To expedite development time, guarantee uniformity across all devices and browsers, and capitalize on its integrated responsive design fea-

tures, developers frequently include Bootstrap into their projects during the development process. Developers can use package managers like npm or yarn to install Bootstrap as a dependency in their project, or they can download the Bootstrap files and link them directly in their HTML files. They can also use build tools like Webpack or Gulp to optimize and compile both their own and Bootstrap code. All things considered, developers choose Bootstrap because it makes the process of building contemporary, responsive web interfaces easier [2].

3.11 Postman

Postman is a multipurpose API testing tool that eases testing, documenting, and sharing of APIs. Its user-friendly environment helps developers to create different HTTP requests and unify requests into collections for easy management. Postman can also create interactive documentation, helps to facilitate collaboration among team members and offers monitoring of multi functionalities for wise-ranging API testing.

3.12 Visual Studio Code

Microsoft provides a free, open-source text editor known as Visual Studio Code (VS Code). It works with Linux, macOS, and Windows. VS Code is popular because it is both powerful and lightweight. It supports various programming languages, including CSS, Go, Java, C++, and Python. Users can install extensions for Visual Studio Code, including tools for debugging, code checking, and cloud and web development support. It is based on the Electron framework, which facilitates the development of cross-platform applications. Visual Studio Code combines a simple editor with capabilities for project management, debugging, and version control (using Git). While Visual Studio 2019 provides a more comprehensive programming experience, VS Code remains quite useful for a variety of jobs. [18].

3.13 GitHub

GitHub is a broadly-used platform for hosting and collaborating on software development of projects using Git version control. Generally, it offers diverse features, such as version control, code hosting, issue tracking and collaboration tools, thus making it a central hub to manage and contribute to projects. Due to its working-friendly interface and vigorous community support, GitHub has become an essential tool for developers all over the world, enabling efficient collaboration, code, sharing, and project management.

4 CRITICAL DESIGN REVIEW AND RESULTS ANALYSIS

This chapter is mainly focusing on the showcase of project designing and all the expected outputs. Also, all the frontend and server-side outputs are presented here by attaching all the relevant pictures. Result analysis and critical design review of the project with proper discussion is presented in this chapter. Generally, this section is the summary of the project outcomes.

4.1 Operating Principle

This section outlines the full operating principle of the proposed inventory management system, detailing its workflows and features, which have been designed to be suitable for various roles within an organization, including employees, branch administrators, and head office administrators. The system is tailored to support these roles by providing streamlined processes for inventory requests, tracking, and approvals.

4.2 User Roles and Dashboards

Before diving into the specific functionalities for each user role, it is important to note that the system is designed to accommodate a wide range of administrative tasks. Below is a breakdown of the features available for each role, ensuring that the workflows support efficient inventory and user management. This table provides a quick overview of the key functions associated with each role.

User Role	Feature	Description
Head Office Admin	User Management	Approve, update, and delete users.
	Product Management	Add, update, delete products, and approve products added by Branch Admins.

	Location Management	Add, update, and delete locations.
	Inventory Management	Add, update, and delete inventory items.
	Assignment Management	Assign products to employees, update assigned products and delete assigned products.
Branch Admin	User Management	Approve, update, and delete branch users.
	Product Management	Add products to the branch inventory and assign them to employees.
	Assignment Management	Assign products to employees and approve product requests made by employees.
Other Employee	Product Management	View products assigned to them, request new products, and transfer products to other employees.
	User Management	Update their own user details.

4.3 Detailed Workflow

4.3.1 Authorization and Authentication Workflow

1. **User Registration**
   - **New User Registration:**
     - **Description:** New users register on the platform by providing necessary details. The registration request is then subject to approval by the Branch Admin.
     - **Steps:**
       - **Visit Registration Page:** The new user navigates to the registration page on the platform.
       - **Fill Registration Form:** The user fills out the registration form with required details such as

name, email, password, role (Branch Admin, Other Employee), and branch location.

- **Submit Form:** Upon submission, the system saves the registration details and marks the user account as "pending approval."

- **Branch Admin Approval**
  - **Description:** The Branch Admin reviews and approves or rejects new user registration requests.
  - **Steps:**
    - **Login to Dashboard:** The Branch Admin logs into their dashboard.
    - **Navigate to User Management:** The Branch Admin goes to the User Management section.
    - **Review Pending Registrations:** The Branch Admin reviews the details of users who have registered and are pending approval.
    - **Approve or Reject Registration:** The Branch Admin approves or rejects the registration request. Approved users receive a confirmation email with login instructions, while rejected users are notified of the rejection.

2. **User Login**

- **Authorized Login**
  - **Description:** Authorized users log in to the platform using their credentials.
  - **Steps:**
    - **Visit Login Page:** The user navigates to the login page.
    - **Enter Credentials:** The user enters their registered email address and password.
    - **Credential Verification:** The system verifies the entered credentials.
    - **Access Granted:** If the credentials are correct, the user is redirected to their respective

dashboard (Head Office Admin, Branch Admin, Other Employee). If incorrect, an error message is displayed, prompting the user to try again.

4.3.2 Head Office Admin Workflow

1. **Login to Head Office Admin Dashboard**
    - **User Management**
        - **Update User Details:**
            - **Description:** Update information for existing users.
            - **Steps:**
                - ✓ **Select User:** Choose a user to edit
                - ✓ **Update Information:** Modify necessary fields.
                - ✓ **Save Changes:** Click "Save" to update the user details
        - **Delete Users:**
            - **Description:** Remove user accounts that are no longer needed
            - **Steps:**
                - ✓ **Select User:** Choose a user to delete
                - ✓ **Confirm Deletion:** Click "Delete" and confirm
    - **Product Management**
        - **Add New Products:**
            - **Description:** Introduce new products to the system.
            - **Steps:**
                - ✓ **Navigate to Product Management:** Click on "Add Product."

- ✓ **Enter Product Details:** Fill in details such as product name, description, price, and category.
- ✓ **Save:** Click "Save" to add the product to the catalog.

- **Edit Product Details:**
  - ➢ **Description:** Modify details of existing products
  - ➢ **Steps:**
    - ✓ **Select Product:** Choose a product to edit
    - ✓ **Update Information:** Modify necessary fields
    - ✓ **Save Changes:** Click "Save" to update the product details
- **Delete Products:**
  - ▪ **Description:** Remove products from the system
  - ▪ **Steps:**
    - ✓ **Select Product:** Choose a product to delete.
    - ✓ **Confirm Deletion:** Click "Delete" and confirm.
- **Approve Products Added by Branch Admin:**
  - ▪ **Description:** Review and approve products submitted by Branch Admins
  - ▪ **Steps:**
    - ✓ **Review Pending Products:** Check the list of pending products.
    - ✓ **Approve or Reject:** Click "Approve" or "Reject" for each product

  - o **Location Management**
- **Add New Locations:**
  - ▪ **Description:** Define new branch locations.
  - ▪ **Steps:**
    - ✓ **Navigate to Location Management:** Click on "Add Location."

- ✓ **Enter Location Details:** Fill in details such as branch name, address, and contact information.
- ✓ **Save:** Click "Save" to add the new location.

- **Edit Location Details:**
  - **Description:** Update information for existing locations.
  - **Steps:**
    - ✓ **Select Location:** Choose a location to edit.
    - ✓ **Update Information:** Modify necessary fields.
    - ✓ **Save Changes:** Click "Save" to update the location details.
- **Delete Locations:**
  - **Description:** Remove locations that are no longer needed
  - **Steps:**
    - ✓ **Select Location:** Choose a location to delete
    - ✓ **Confirm Deletion:** Click "Delete" and confirm

o **Inventory Management**

- **Add New Inventory Items:**
  - **Description:** Add new items to the company's central inventory
  - **Steps:**
    - ✓ **Navigate to Inventory** Management: Click on "Add Inventory Item"
    - ✓ **Enter Item Details:** Fill in details such as item name, description, quantity, and storage location
    - ✓ **Save:** Click "Save" to add the new inventory item
- **Edit Inventory Item Details:**
  - **Description:** Update details for existing inventory items
  - **Steps:**
    - ✓ **Select Item:** Choose an item to edit
    - ✓ **Update Information:** Modify necessary fields
    - ✓ **Save Changes:** Click "Save" to update the inventory item details
- **Delete Inventory Items:**

- **Description:** Remove inventory items that are no longer needed
- **Steps:**
  - ✓ **Select Item:** Choose an item to delete
  - ✓ **Confirm Deletion:** Click "Delete" and confirm

4.4 Branch Admin Workflow

1. **Login to Branch Admin Dashboard**
   - **User Management**

- **Approve Registered Users for the Branch**
  - **Description:** Review and approve or reject new user registrations for the branch
  - **Steps:**
    - ✓ **Navigate to User Management:** Review the list of pending registrations
    - ✓ **Approve or Reject:** Click "Approve" or "Reject" for each user

  - **Product Management**

- **Add Products to Branch Inventory:**
  - **Description:** Add products to the branch inventory, pending Head Office Admin approval
  - **Steps:**
    - ✓ **Navigate to Product Management:** Click on "Add Product"
    - ✓ **Enter Product Details:** Fill in details such as product name, description, and quantity
    - ✓ **Submit for Approval:** Click "Submit" to send the product details to the Head Office Admin for approval
- **After Approval, Add Products to Branch Inventory:**
  - **Description:** Once products are approved by the Head Office Admin, add them to the branch inventory
  - **Steps:**

- ✓ **Add to Inventory:** Navigate to the product management section, select the approved product, and add it to the branch inventory

- **Assign Products to Employees:**
  - **Description:** Distribute products to employees within the branch
  - **Steps:**
  - **Select Product:** Choose a product to assign
  - **Select Employee:** Choose an employee to assign the product to
  - **Record Assignment:** Enter assignment details and save

- **Assignment Management**

- **Assign Products to Employees:**
  - **Description:** Directly assign products to employees
  - **Steps:**
    - ✓ **Navigate to Assignment Management:** Click on "Assign Product"
    - ✓ **Select Employee and Product:** Choose the employee and product
    - ✓ **Record Assignment:** Enter assignment details and save
- **Approve Product Requests:**
  - **Description:** Review and approve or reject product requests made by employees
  - **Steps:**
    - ✓ **Review Requests:** Navigate to the list of pending product requests
    - ✓ **Approve or Reject:** Click "Approve" or "Reject" for each request
    - ✓ **Notify Employee:** Approved requests result in products being assigned to employees

4.5 Other Employee Workflow

1. **Login to Employee Dashboard**

- **Product Management**

- **View Assigned Products:**
  - **Description:** Employees can view products that have been assigned to them
  - **Steps:**
    - ✓ **Navigate to Product Management:** View the list of assigned products.
- **Request New Products:**
  - **Description:** Employees can request new products from the branch inventory
  - **Steps:**
    - ✓ **Click "Request Product":** Enter request details and submit
    - ✓ **Await Approval:** Wait for the Branch Admin to approve or reject the request
- **Transfer or Handover Products:**
  - **Description:** Employees can transfer or handover products to other employees
  - **Steps:**
    - ✓ **Select Product:** Choose a product to transfer
    - ✓ **Select Employee:** Choose the recipient employee
    - ✓ **Confirm Transfer:** Confirm the transfer and record details in the system

4.6 Process Flow Summary

1. **User Registration and Approval**
   - **New User Registration:** Users register on the platform, filling out necessary details and awaiting approval
   - **Branch Admin Approval:** Branch Admins review and approve or reject new user registrations
2. **User Login and Authorization**
   - **Login: Users log in using authorized credentials.**
3. **Adding and Approving Products**

- **Branch Admin:** Adds products to inventory and submits for approval
- **Head Office Admin:** Reviews and approves products. Approved products are added to the system inventory and Branch Admin is notified
- **Branch Admin:** Manages approved products and assigns them to employees

4. **User Management**
   - **Users:** Register on the platform and await approval
   - **Branch Admin:** Reviews and approves user registrations
   - **Head Office Admin:** Manages user roles and permissions
5. **Product Requests and Transfers**
   - **Employees:** Request products through their dashboard
   - **Branch Admin:** Reviews and approves product requests, assigns approved products to employees
   - **Employees:** Transfer products to other employees, with all transfers recorded and new assignments updated in the system

4.7 Results Analysis

In this section all the corresponding pictures of the outputs from frontend, backend and server-side presented.

4.7.1 Analysing Response Data from Server-side APIs

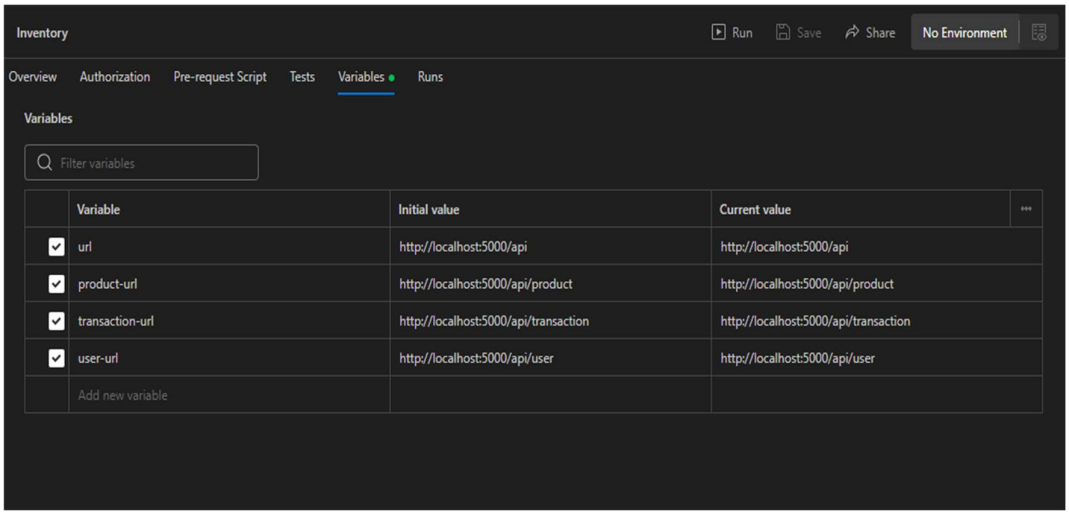


Figure 5. Creation of variables

In figure 5, The image shows the "Variables" tab of a Postman environment configuration for an inventory-related project. There are four variables defined, each representing different API endpoints:

1. **url**: The base URL for the API, set to http://localhost:5000/api.
2. **product-url**: The endpoint for product-related requests, set to http://localhost:5000/api/product.
3. **transaction-url**: The endpoint for transaction-related requests, set to http://localhost:5000/api/transaction.
4. **user-url**: The endpoint for user-related requests, set to http://localhost:5000/api/user.

Both the "Initial value" and the "Current value" are set to the same URLs, indicating that no changes have been made to these variables since they were initialized.

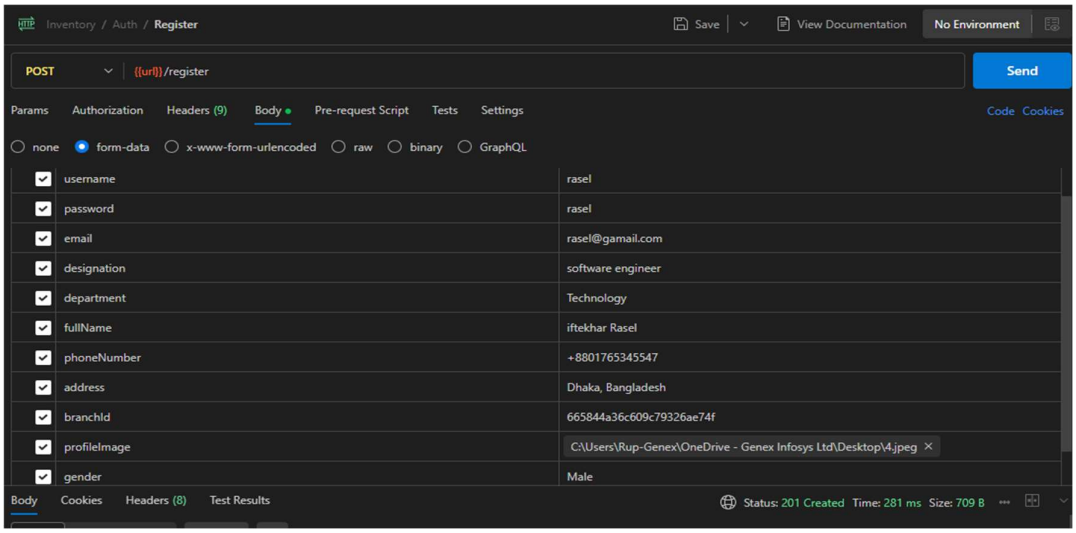


Figure 6: Registration of user, using the post method form

In the figure 6, The provided image displays a **POST** request in Postman for a user registration process, targeting the /register endpoint under the Inventory/Auth system. The request is configured to send **form-data**, which includes a mix of text fields and a file upload, indicative of a typical user profile creation flow. Key information such as the username, password, email, designation, department, and personal details including full name, phone number, address, and gender are included in the form data. Additionally, a branchId is provided, likely identifying the user's office branch. While the form data includes a plain-text password, once submitted, the server will hash the password using a secure hashing bcrypt algorithm before storing it in the database. The request also includes a profile image file, uploaded from a local path, highlighting that the system allows users to associate images with their profiles. The data presented in the image is fictional and does not represent real users or information.

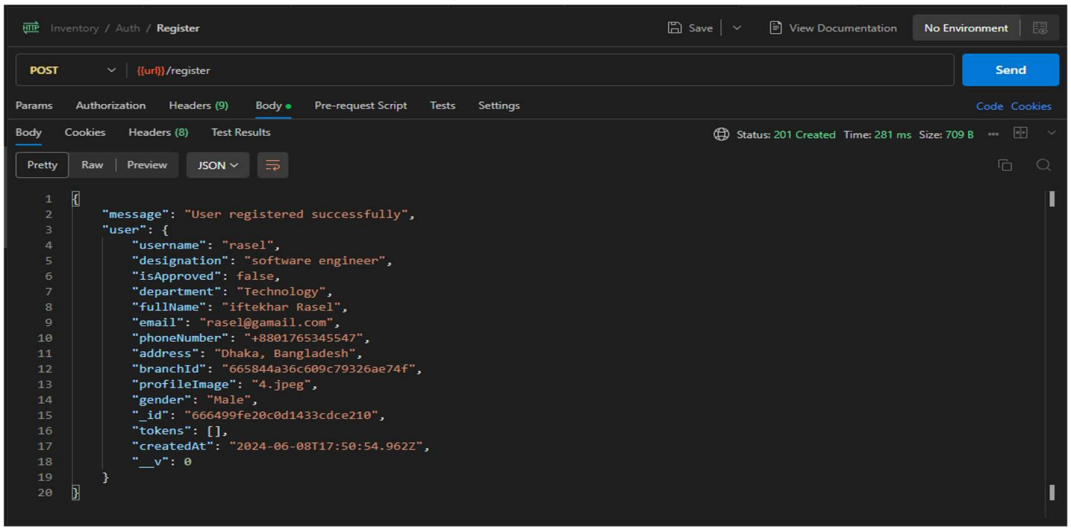


Figure 7. Payload of the register from the register API

In figure 7, The image shows a JSON response from a successful user registration request, with a message confirming that the user has been registered. The response contains user details such as the username, designation, department, full name , email, phone number, address, branch ID, profile image filename, and gender. The user's approval status is marked as "false," indicating that further approval is required before they can access their account. The HTTP status code "201 Created" confirms the successful creation of the user resource.

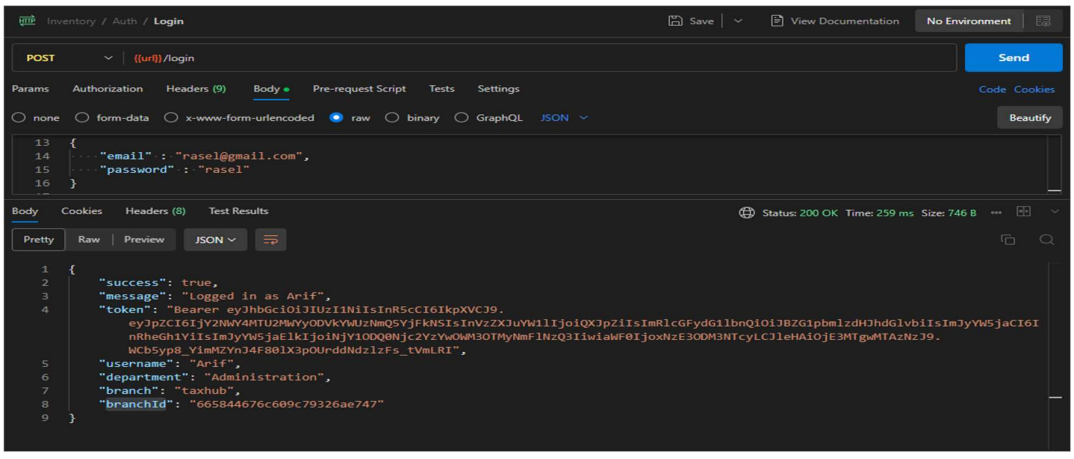


Figure 8. Successful API Login Response

Figure 8 describes a successful API login response in Postman. The request is a POST method to the endpoint /login with JSON body parameters including email and password. The response is shown as JSON, indicating a status code 200 OK. The returned data confirms a successful login with "success": true and a

message. The response also includes an authorization token under "token", along with user details such as username, department, branch, and a branchId.

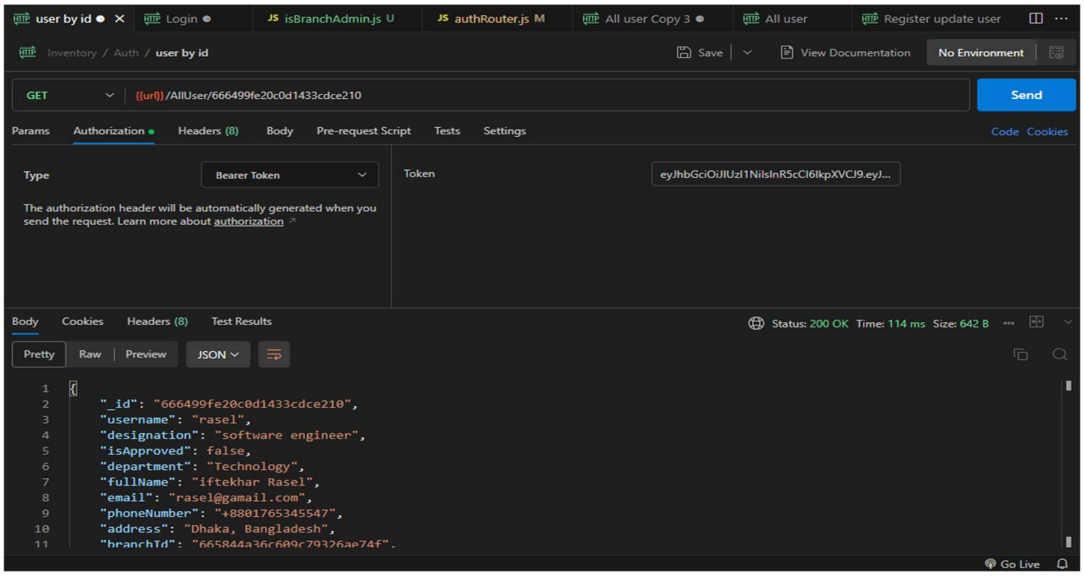


Figure 9. Get method API for showing user information

Figure 9 demonstrates a GET method API request in Postman for retrieving user details based on the user ID provided in the URL (/AllUser/666499fe20c0d1433cdce210). The request uses a Bearer Token for authorization. The response, with a status of 200 OK, displays user information stored in the database, including fields such as username, designation, email, and more. This API is used to fetch details of registered users, and administrators can view and approve pending users. The endpoint is restricted to the head office and branch offices administration departments for managing user approvals.

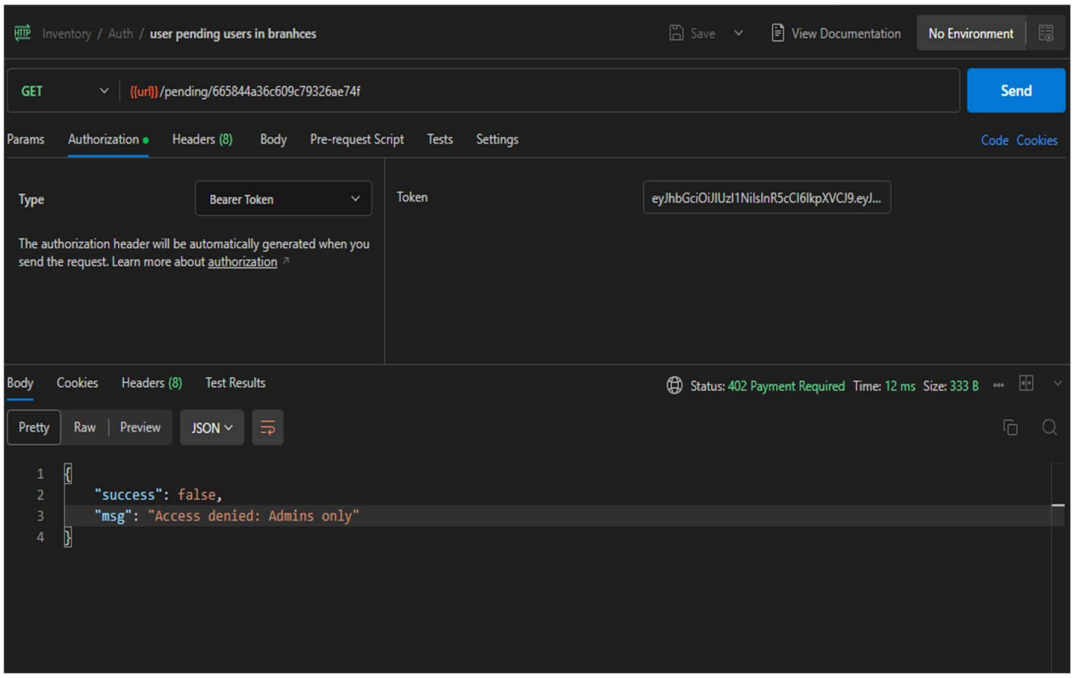


Figure 10. Admin interface for pending registered user

In figure 10, The image depicts a failed API request. The request is a GET request to the URL “/pending/665844a36c609c79326ae74f". The request is authorized using a Bearer token. The response from the server with the message "Access denied: Admins only".

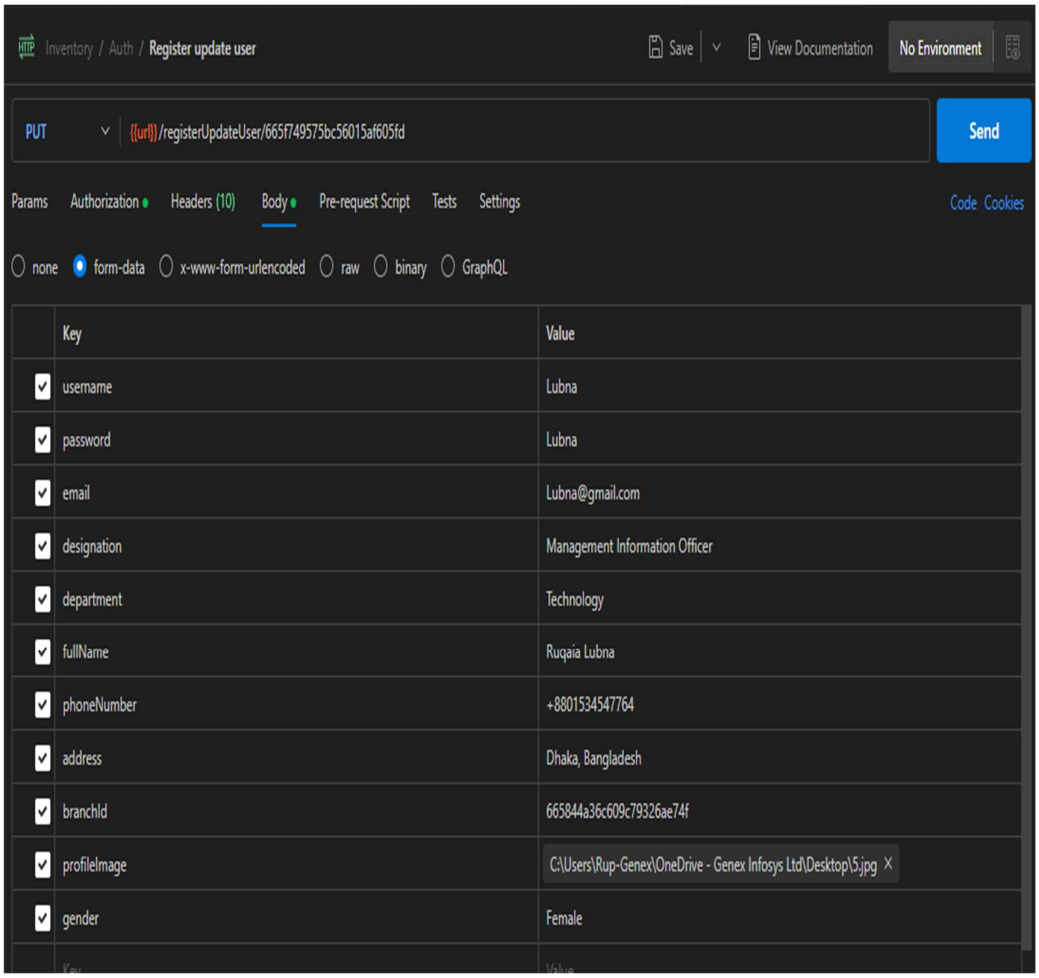


Figure 11. Put method URL for updating the registered users

In the figure 11, form data depicts a PUT request intended to update a user's information within a specific system or application. The request targets the end-point
/registerUpdateUser/665f749575bc56015af605fd,
where 665f749575bc56015af605fd is the user's unique identifier.
The request includes a set of key-value pairs representing the user's details, using their user ID. These details encompass personal information such as username, password, email, full name, and gender, as well as professional information including designation, department, address, and branch affiliation. This API is used for both user updates and new user approvals. Only branch administrators and head office administrators have the authority to approve new users.

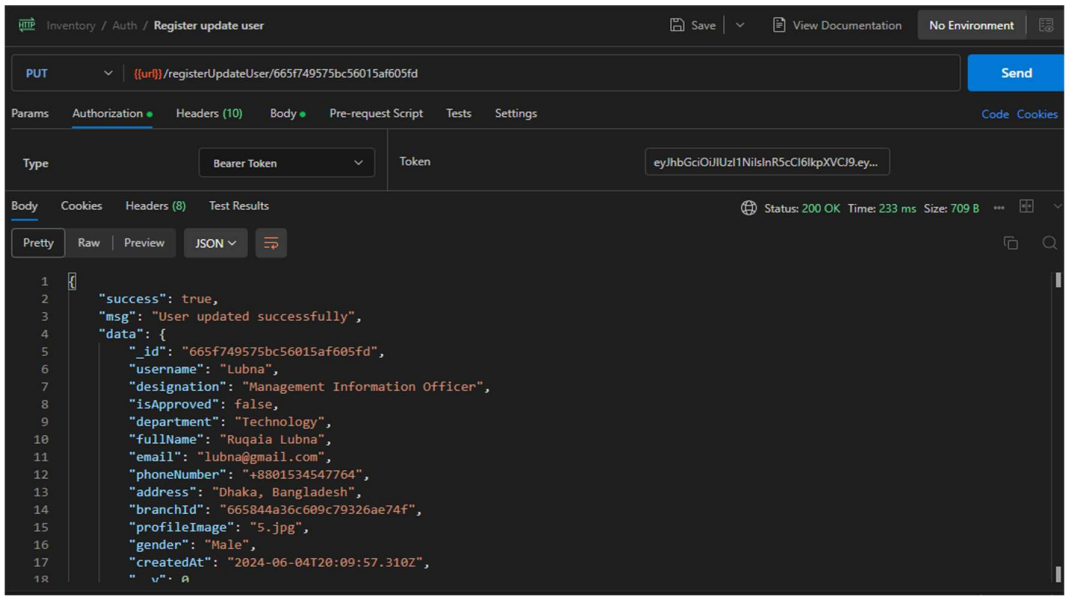


Figure 12. Response json payload of the update user URL

In the figure 12 showcases the successful response to a PUT request sent to the /registerUpdateUser/665f749575bc56015af605fd endpoint, confirming that the user's information has been successfully updated. The response details include the user's unique identifier, username, designation, approval status, department, full name, email address, phone number, address, branch affiliation, profile image, gender, and creation timestamp. Notably, the isApproved field indicates that the user has not yet been approved. If the user has been approved, the value of the isApproved field would be set to "true." This indicates that the user's registration has been completed and they are now authorized.

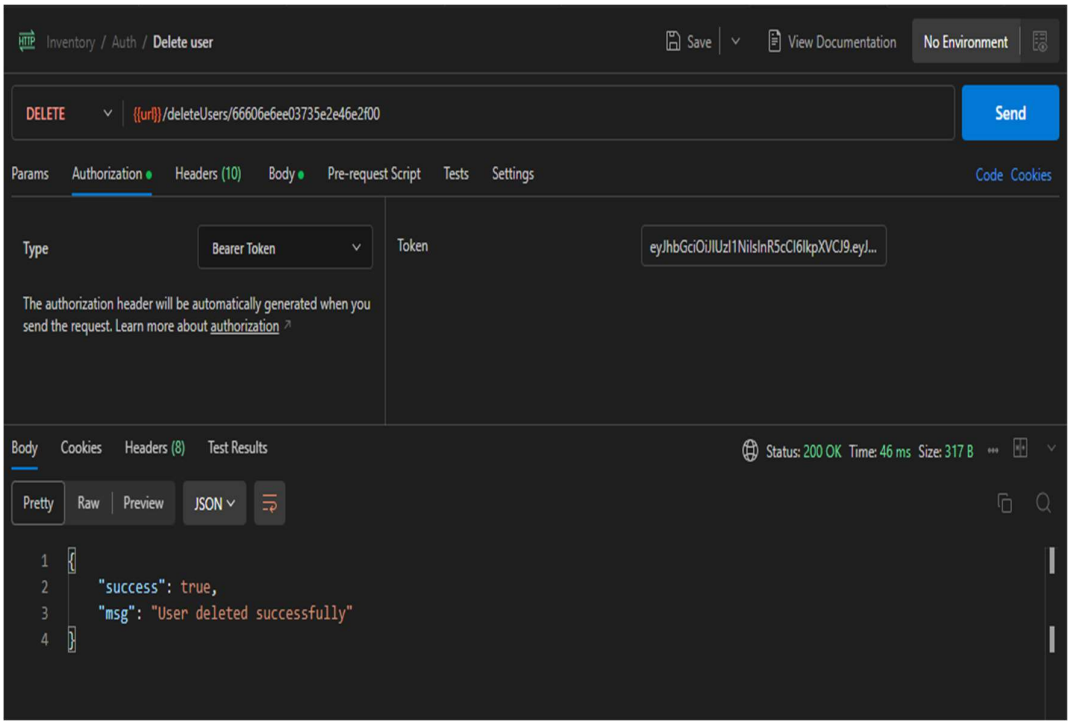


Figure 13. Delete method API for deleting a user

Figure 13 shows a request to delete a user. The request is a DELETE request to the endpoint /deleteUsers/66606e6ee03735e2e46e2f00.

The 66606e6ee03735e2e46e2f00 part of the endpoint is the user's ID, which uniquely identifies the user. This ID is used to locate and delete the user's information from the database. The request includes a bearer token in the authorization header. The response is a 200 OK status code with the message "User deleted successfully". This indicates that the user was successfully deleted from the system.

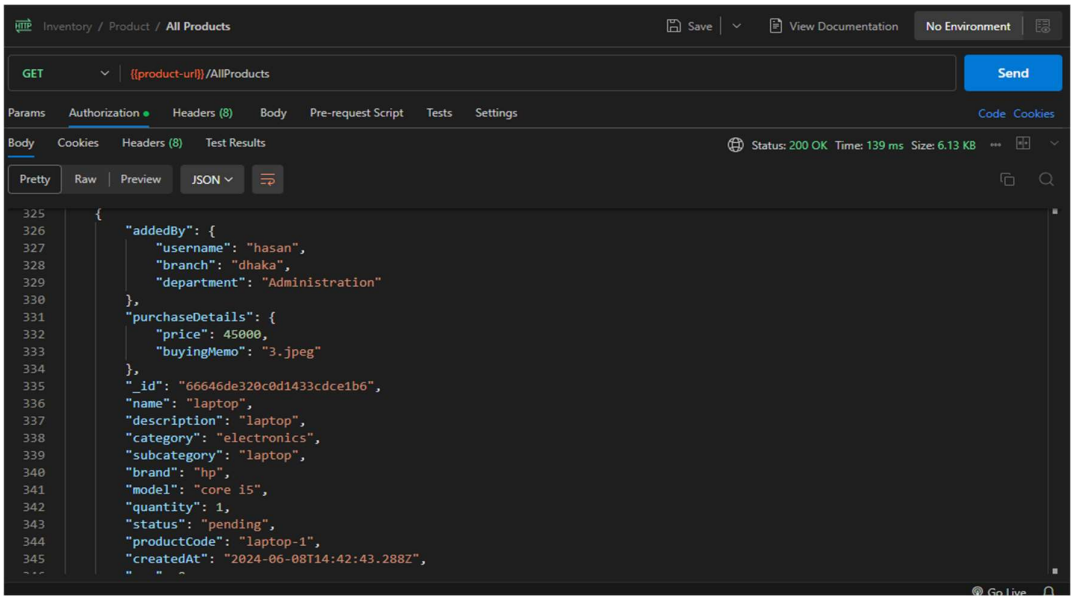


Figure 14. Get method for showing added product in the central inventory

In figure 14 depicts a Postman request aimed at retrieving a comprehensive list of all products within a specified system or application. The GET request targets the endpoint {{product-url}}/AllProducts, resulting in a successful 200 OK response that includes a JSON object containing detailed product information. Each product entry within the response encompasses a unique identifier, name, description, categorization details, brand, model, quantity, status, product code, and creation timestamp. Additionally, the response includes information about the user who added the product and purchase-related details such as price and a buying memo. This comprehensive data provides a overview of the available products and their associated attributes within database.

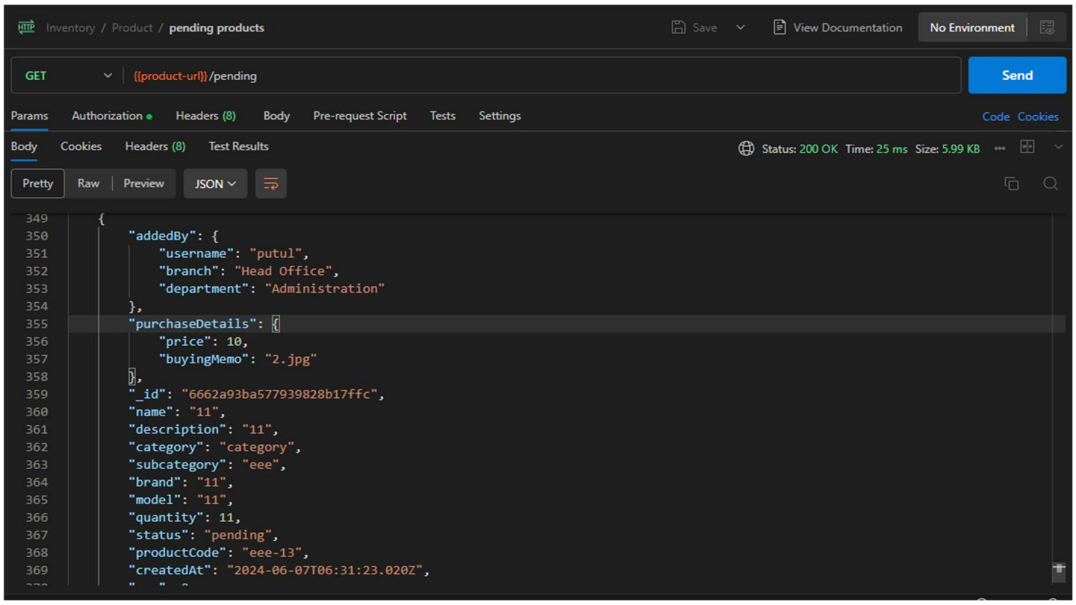


Figure 15. Pending products API

Figure 15 show that the system has a two-step approval process. Branch administrators are responsible for adding new products to the system, but these products remain in a pending state until they are approved by head office administrators. The {{product-url}}/pending endpoint likely retrieves a list of all products that are currently awaiting approval. Once a product is approved by a head office administrator, it will be moved to an approved or active state, and will no longer appear in the pending list.

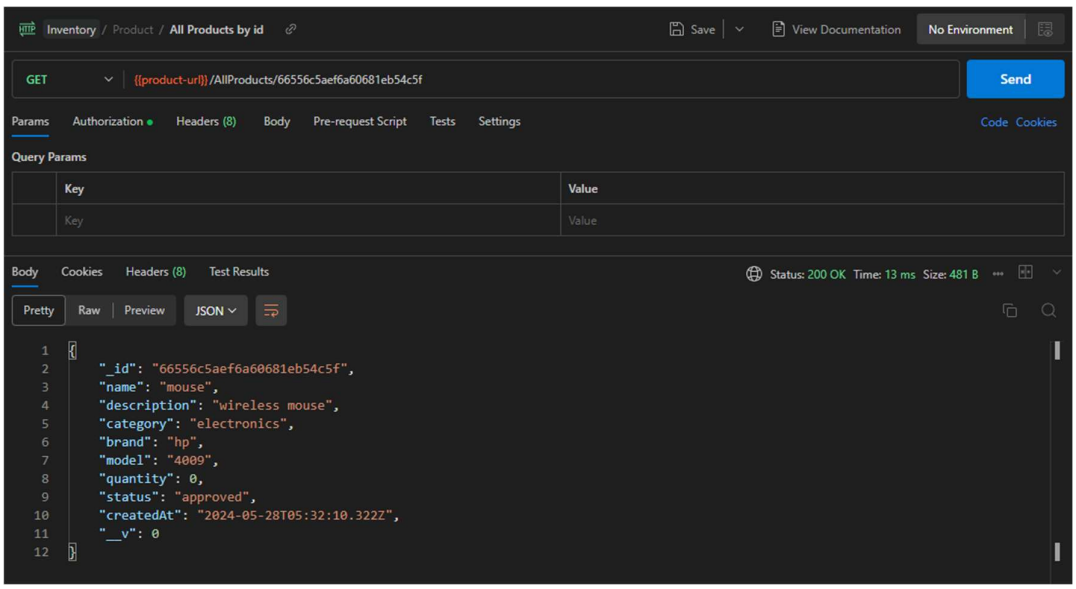


Figure 16. User data API

Figure 16 shows a request aimed at retrieving a specific product from database. The GET request targets the endpoint /{{product-url}}/AllProducts/66556c5aef6a60681eb54c5f, where the 66556c5aef6a60681eb54c5f segment serves as a unique identifier for that specific product. The successful 200 OK response includes a JSON object containing detailed product information, encompassing its unique identifier, name, description, categorization details, brand, model, quantity, status, and creation timestamp.

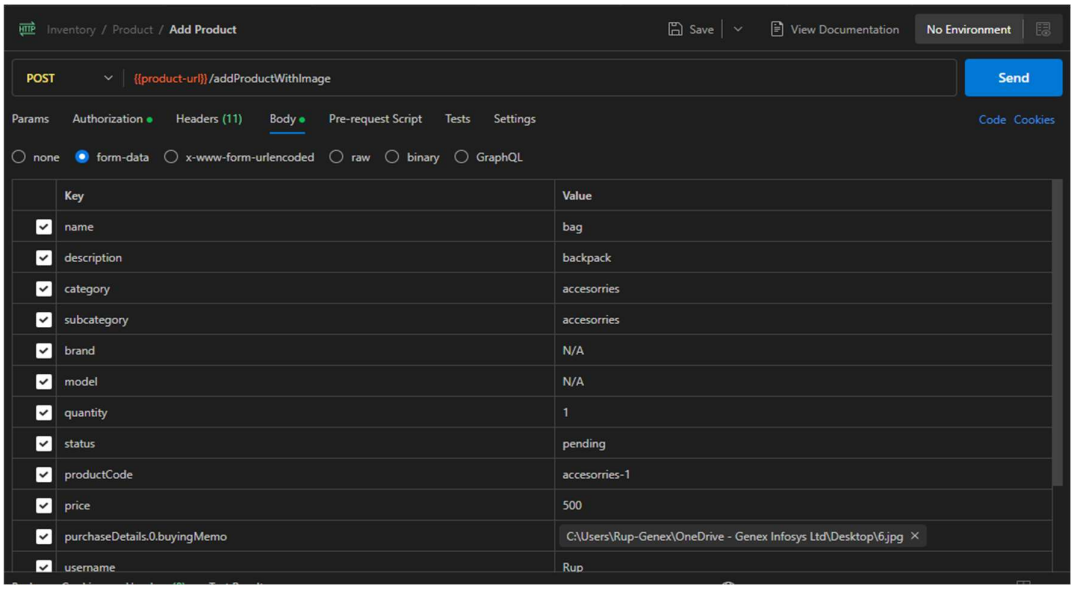


Figure 17. API to add any product to the inventory

Figure 17 shows that a request aimed at adding a new product to a system. The POST request targets the endpoint /{{product-url}}/addProductWithImage, accompanied by a set of key-value pairs representing the product details. These details include the product's name, description, categorization information, brand, model, quantity, status, product code, price, and an attached image file. Upon successful submission, the request will likely result in the creation of a new product entry within the system, pending approval by head office administrators.

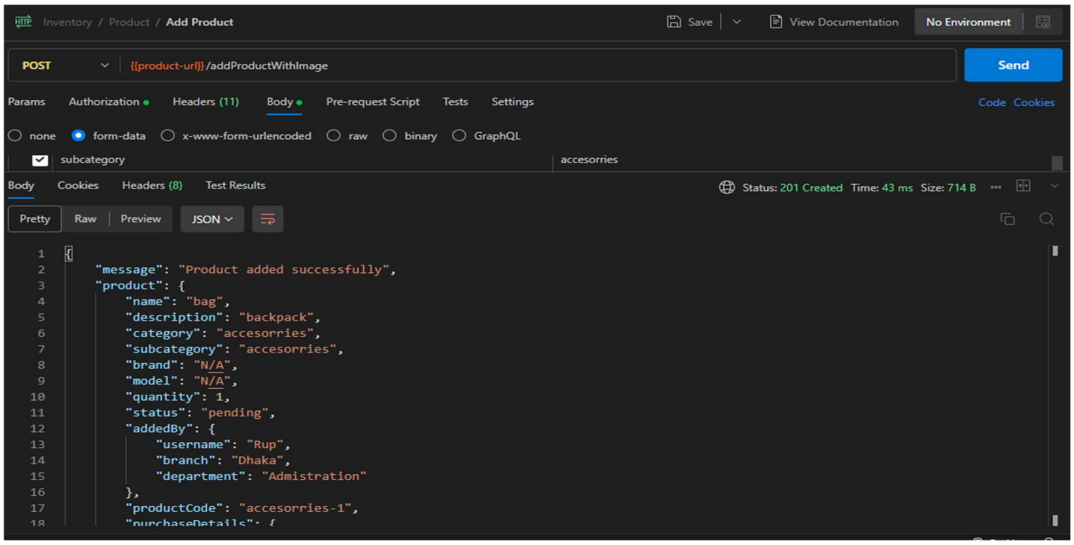


Figure 18. Response data of the add product API

Figure 18 showcases the successful response to POST request sent to the /{{product-url}}/addProductWithImage endpoint, confirming that the new product has been added to the system. The response message indicates that the product was added successfully, and the response body includes details of the newly created product, such as its unique identifier, name, description, categorization information, brand, model, quantity, status, product code, and creation timestamp. Notably, the status field is set to "pending," indicating that the product requires further approval from head office administrator before it becomes active within the system.

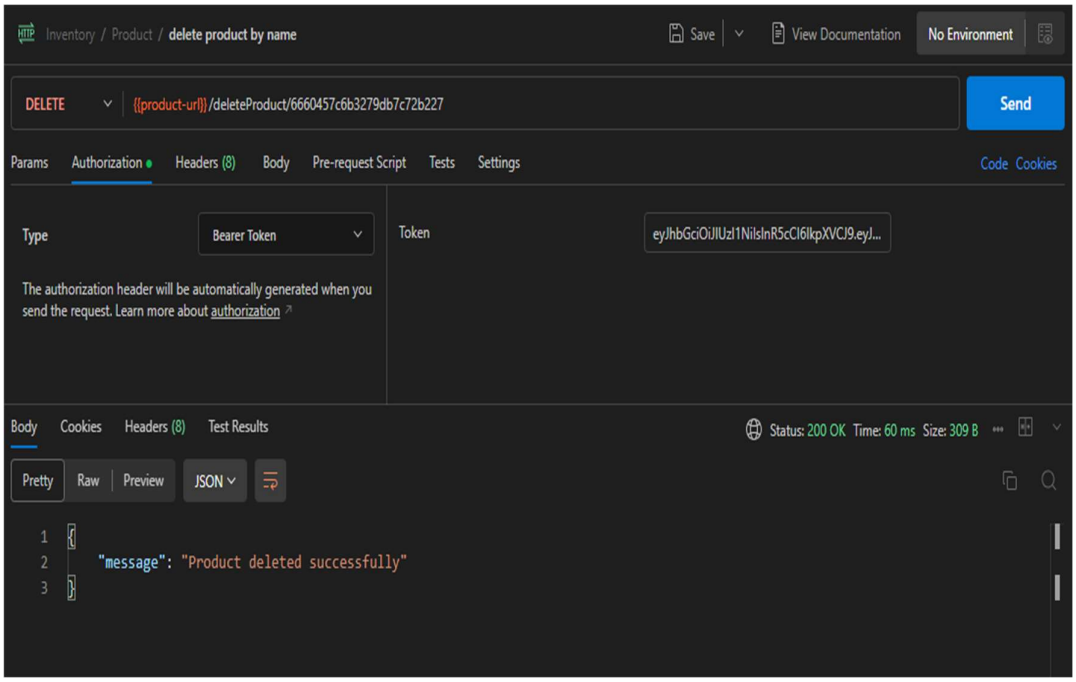


Figure 19. Delete method for deleting any product

Figure 19 shows a request aimed at deleting a specific product from a specified system. The DELETE request targets the endpoint /{{product-url}}/delete-Product/6660457c6b3279db7c726227, where the 6660457c6b3279db7c726227 segment serves as a unique identifier for the product to be deleted. Upon successful execution, the request will likely result in the removal of the specified product from the system, as indicated by the "Product deleted successfully" message in the response

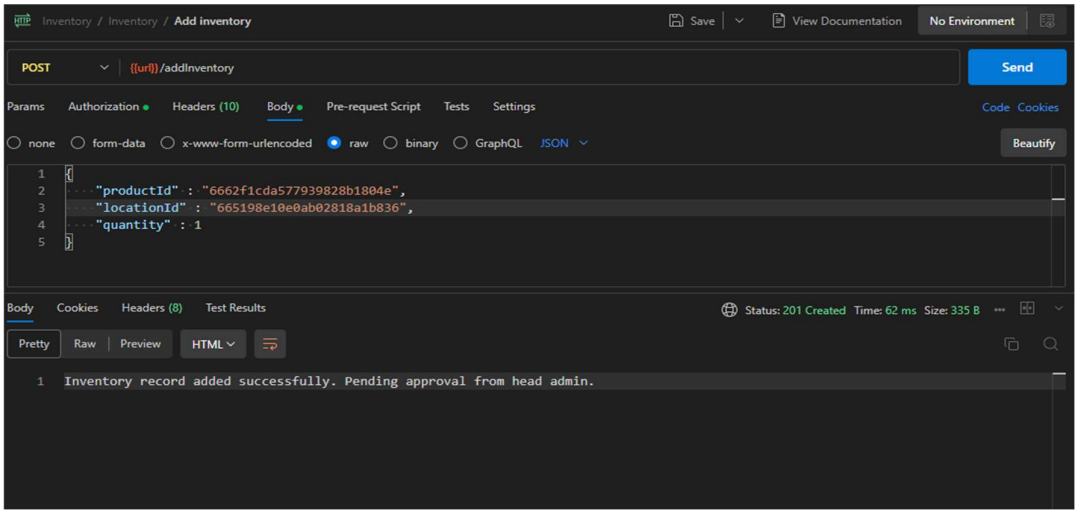


Figure 20. Post method for adding products in any specific branch

Figure 20 shows a request aimed at adding a new inventory item to a specified branch within the system. The POST request targets the endpoint /{{inventory}}/addInventory, accompanied by the necessary product and branch information, including the product ID, location ID, and quantity. Upon successful submission, the request will likely result in the creation of a new inventory record within the system, pending approval by the head administrator. The product ID and location ID are unique identifiers assigned to the respective product and location, enabling accurate management of inventory items within the system.

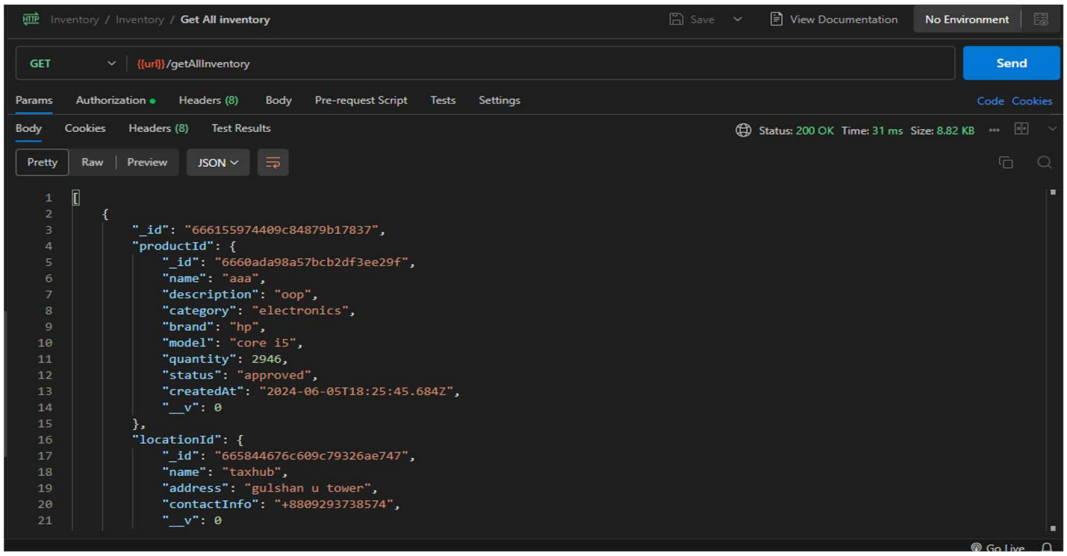


Figure 21. Get method for getting all the inventory

Figure 21 shows a request aimed at retrieving all inventory items from a specified system or application. The GET request targets the endpoint /getAllInventory, resulting in a successful 200 OK response that includes a JSON object containing a list of inventory items.

Each inventory item within the response includes a unique identifier, product details (including product ID, name, description, category, brand, model, quantity, status, and creation timestamp etc.), and location details (including location ID, name, address, and contact information etc.). This comprehensive data provides a overview of the inventory within the system, including information about product quantities, locations.

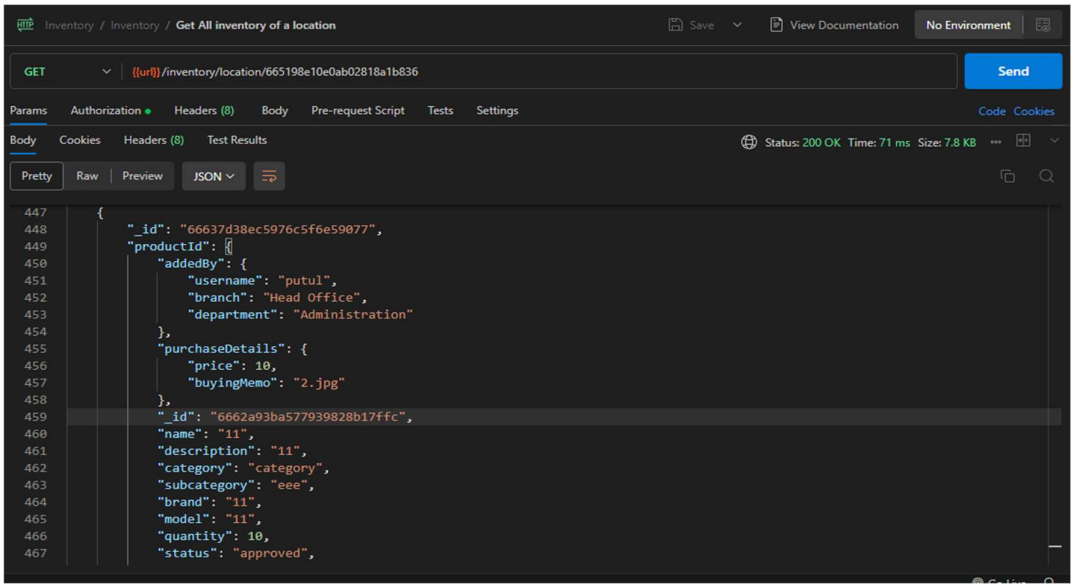


Figure 22: get method to display specific location's product details

Figure 22 demonstrates a GET method that retrieves detailed information about all products available at a specific location. This functionality is accessible to both branch administration department personnel and head office administration department personnel. By utilizing this method, users can effectively determine which products are currently in stock at a particular branch.

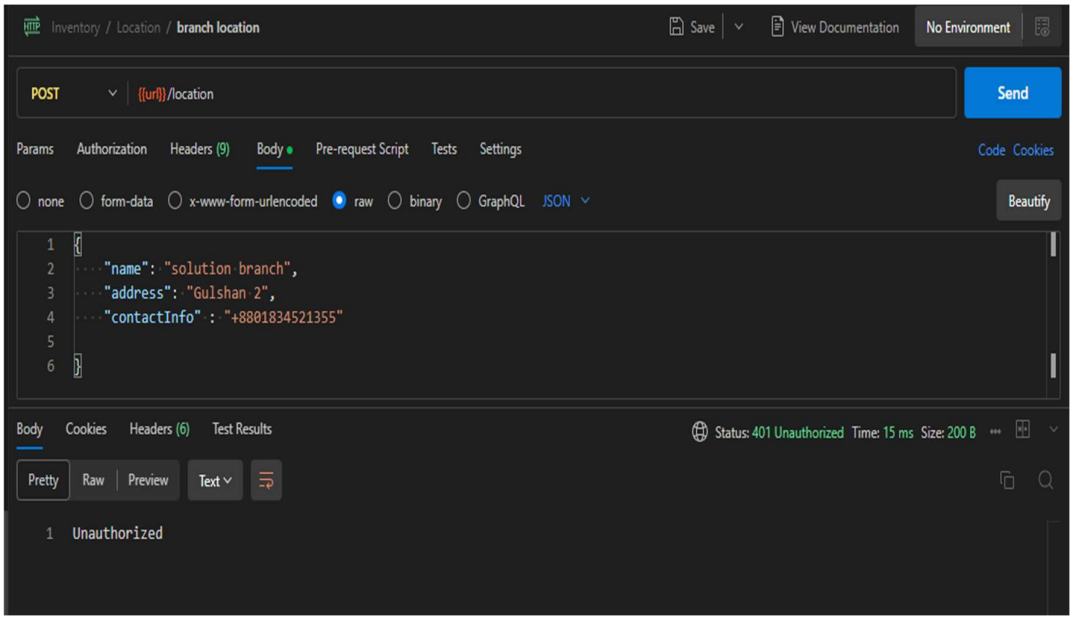


Figure 23. Post method to add new location

Figure 23 illustrates a request designed to create a new branch location within system. The request includes the necessary branch information, such as the name, address, and contact details. However, the request is likely unauthorized as only head office administrators have the authority to add new branch locations.

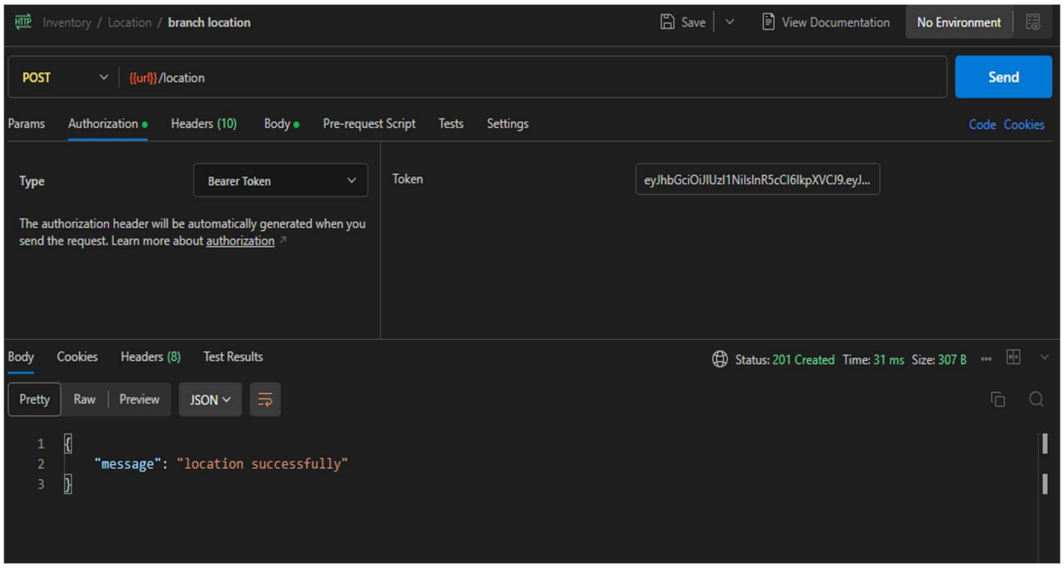


Figure 24. Post method to add new location (Authorize head branch admin)

Figure 24 illustrates a request designed aimed at adding a new branch location to a specified system. The request includes the necessary branch information, such as the name, address, and contact details. However, only head office administrators are authorized to add new branch locations. The response indicates that the location was created successfully, likely due to the presence of a valid authentication token in the request header, verifying the user's credentials as a head office administrator.

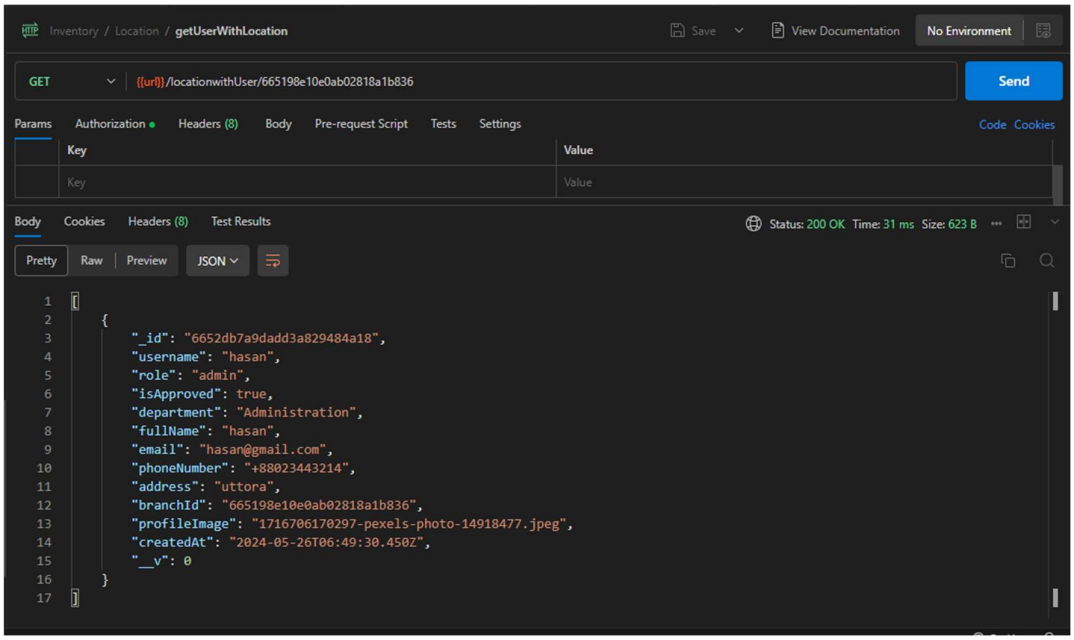


Figure 25. API to show all the user/employees of branch

Figure 25 depicts a GET method request designed to retrieve a list of all users associated with a specific branch within system. The branch ID is incorporated into the URL of the request, allowing for targeted retrieval of employee information. The response from this API will include the branch ID of the requested branch, along with detailed information about each employee affiliated with that branch.

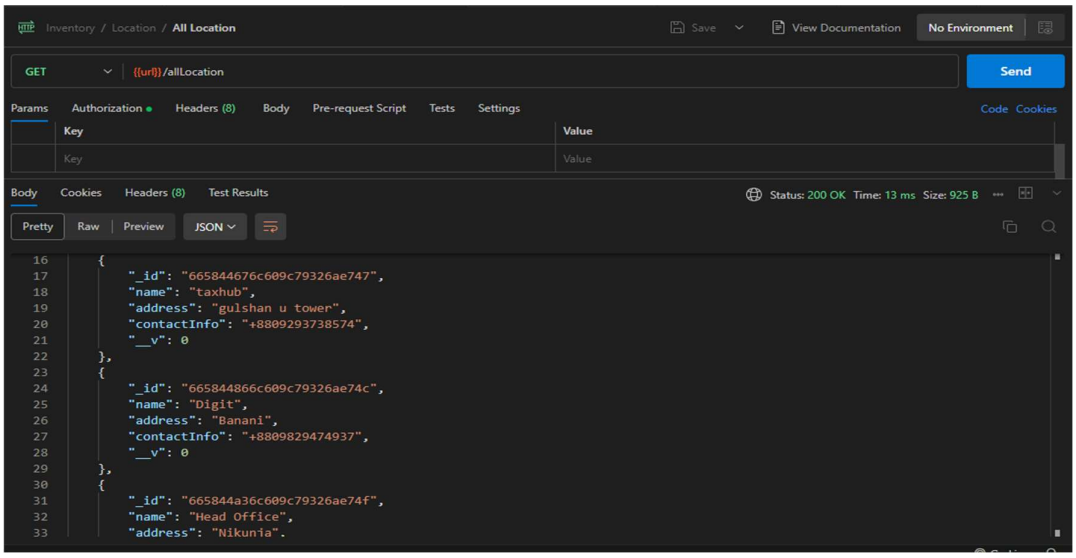


Figure 26. API for showing the locations

Figure 26 shows a request aimed at retrieving a list of all locations. The GET request targets the endpoint /getAllLocation, resulting in a successful 200 OK response that includes a JSON object containing a list of locations.

Each location within the response includes a unique identifier, name, address, and contact information. This comprehensive data provides a overview of the available locations within the system.

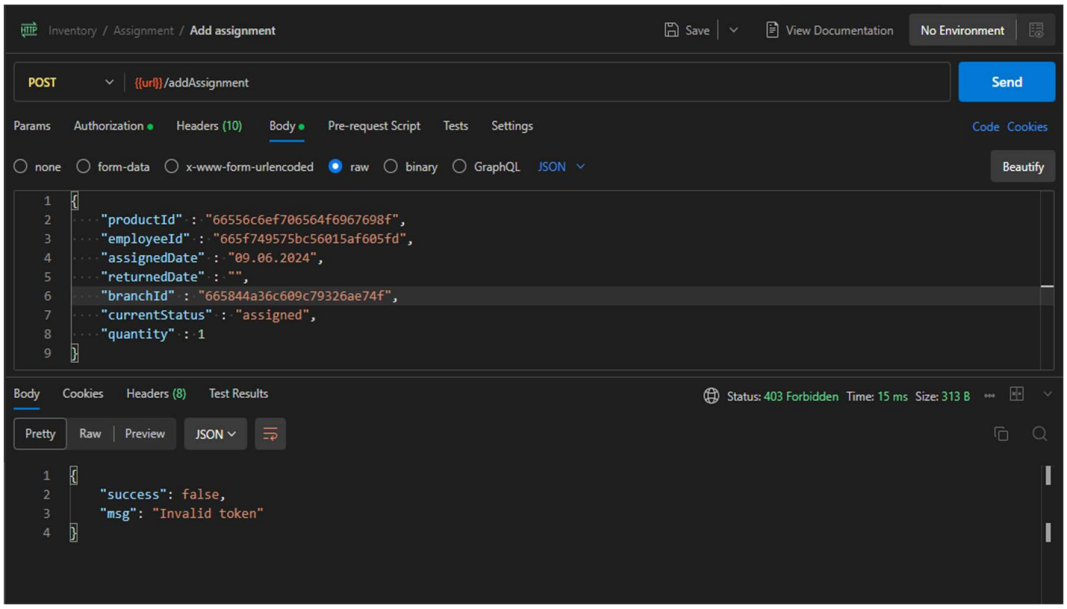


Figure 27. Assigning of products to employees

In Figure 27, it illustrates a request designed to assign a specific product to an employee within specified system. The request includes the necessary product and employee information, such as the product ID, employee ID, assigned date, returned date, branch ID, current status, and quantity. Upon successful submission, the request will result in the creation of a new assignment record within the system, associating the specified product with the designated employee. This functionality is restricted to branch or head office administration department personnel.

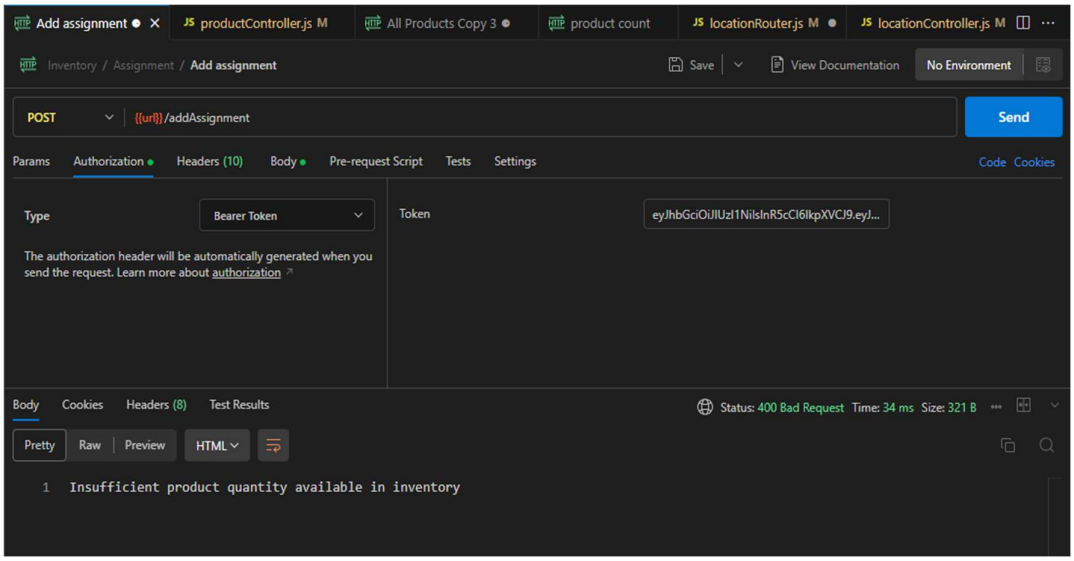


Figure 28. Response data for insufficient product quantity

In Figure 28, the request resulted in a 400 Bad Request error with the message "Insufficient product quantity available in inventory." This indicates that there is not enough of the specified product in the inventory at the assigned branch location to fulfill the assignment request.

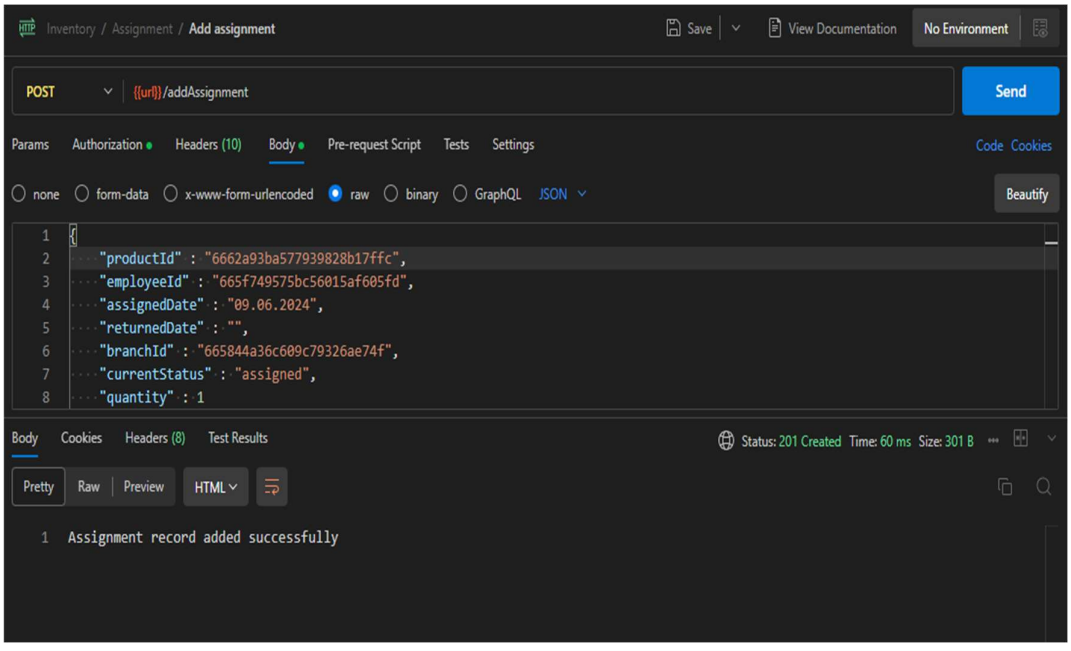


Figure 29. Product assigning to belongings branch employee

Figure 29 shows that the request was successful, with a 201 Created status code and a message indicating that the assignment record was added successfully. This suggests that the product was successfully assigned to the employee, and the assignment details were recorded in the system.

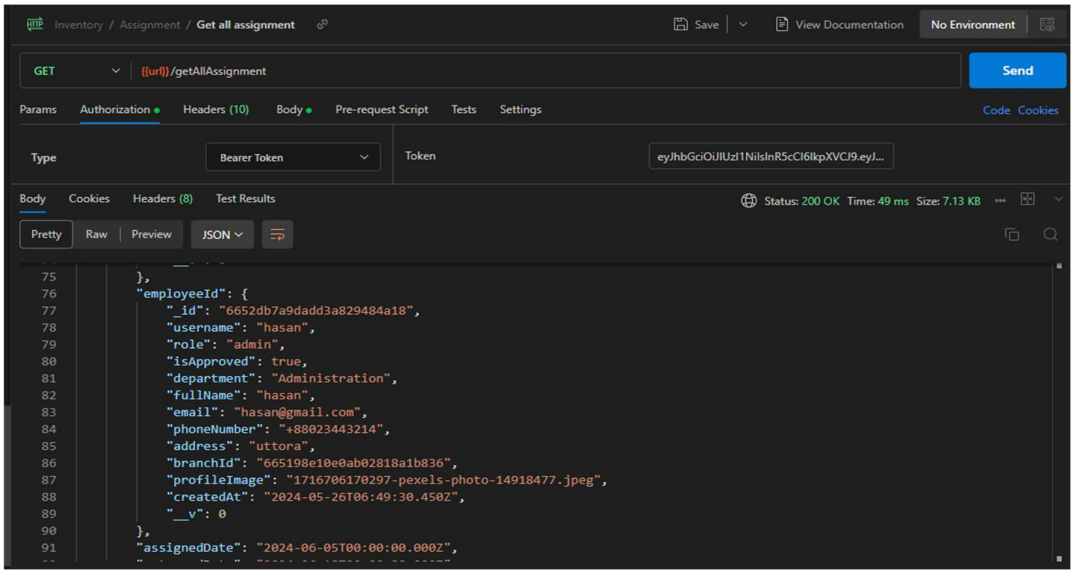


Figure 30. Get method to show assigned products to the employee

Figure 30 is giving a clear view of GET request targets the endpoint /getAllAssignment, resulting in a successful 200 OK response that includes a JSON object containing a list of assignments.

Each assignment within the response includes a unique identifier, employee details (including employee ID, username, role, approval status, department, full name, email, phone number, address, branch ID, profile image, and creation timestamp), assigned date, and other relevant information. This comprehensive data provides a overview of all assignments within the system, including information about assigned products, employees, and assignment dates.

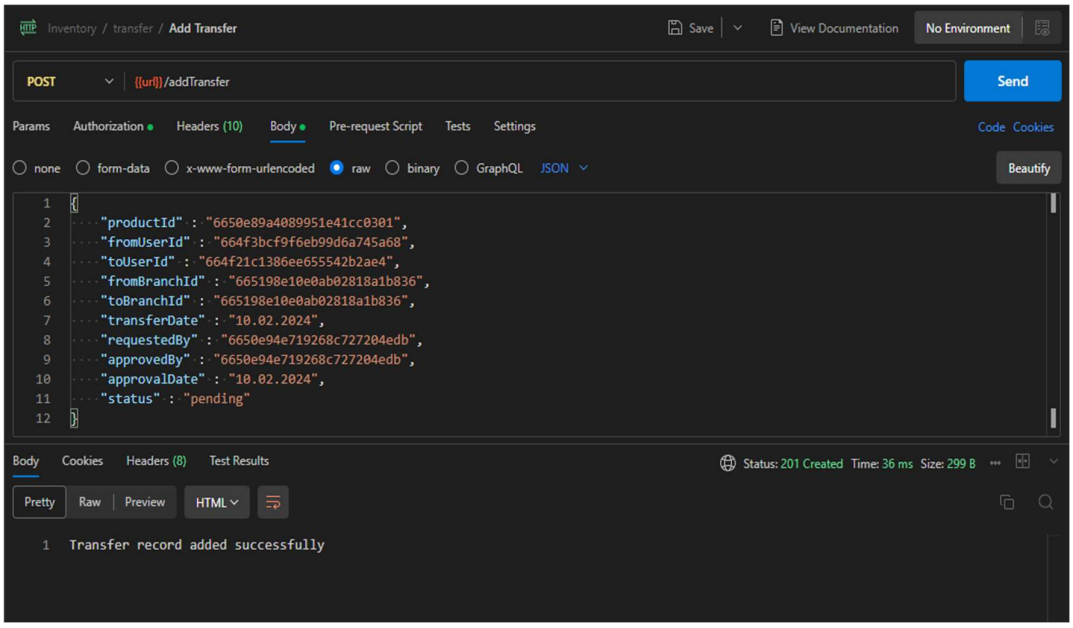


Figure 31. Post method to transfer a product to the other employee

In figure 31, The POST request targets the endpoint /addTransfer, accompanied by the necessary transfer information, including the product ID, transferring user ID, receiving user ID, transferring branch ID, receiving branch ID, transfer date, requested by user ID, approved by user ID, approval date, and status. Upon successful submission, the request will likely result in the creation of a new transfer record within the system, initiating the process of transferring the specified product from one location to another. This functionality is restricted to authorized users within the system.

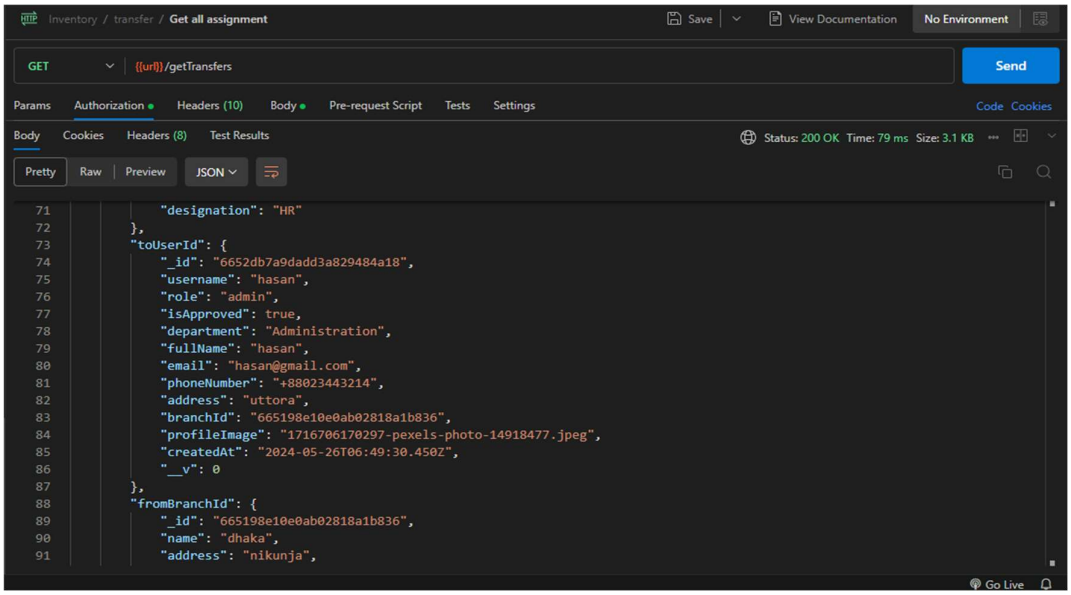


Figure 32. API to show transfer of any product

In figure 32, The GET request targets the endpoint /getAllTransfers, resulting in a successful 200 OK response that includes a JSON object containing a list of transfers.

Each transfer within the response includes a unique identifier, product details (including product ID, name, description, category, brand, model, quantity, status, and creation timestamp), transferring user details, receiving user details, transferring branch details, receiving branch details, transfer date, requested by user details, approved by user details, approval date, and status. This comprehensive data provides a overview of all inventory transfers within the system, including information about transferred products, locations, dates, and approval status.

4.7.2 Analysing Frontend Response

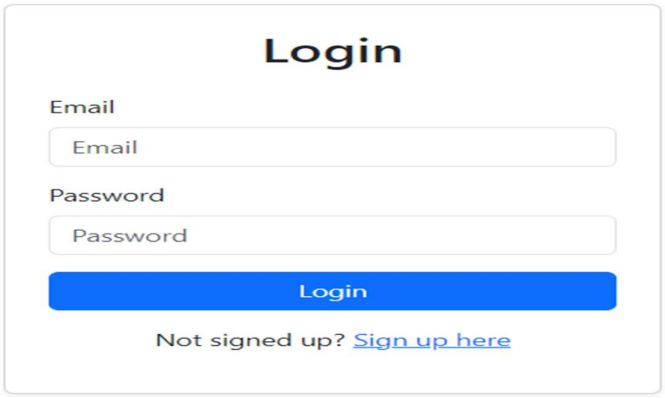


Figure 33. Login environment

Figure 33 shows the log in page. Only authorized personnel and approved users can log in through this page.

Figure 34. Registration page

Figure 34 shows the registration page where a form requires users to input a variety of personal and professional details, including their username, password,

email address, full name, gender, designation, department, address, branch affiliation, and profile image. Upon entering the necessary information, users can submit the form to create a new account. The form layout is simple and easy to navigate, with clearly labeled fields and a prominent "Register" button. The system includes validation rules to ensure the accuracy and completeness of the entered data. Upon successful submission, the system will process the information and create a new user account, but the user will need to wait for their account to be approved before they can log in.

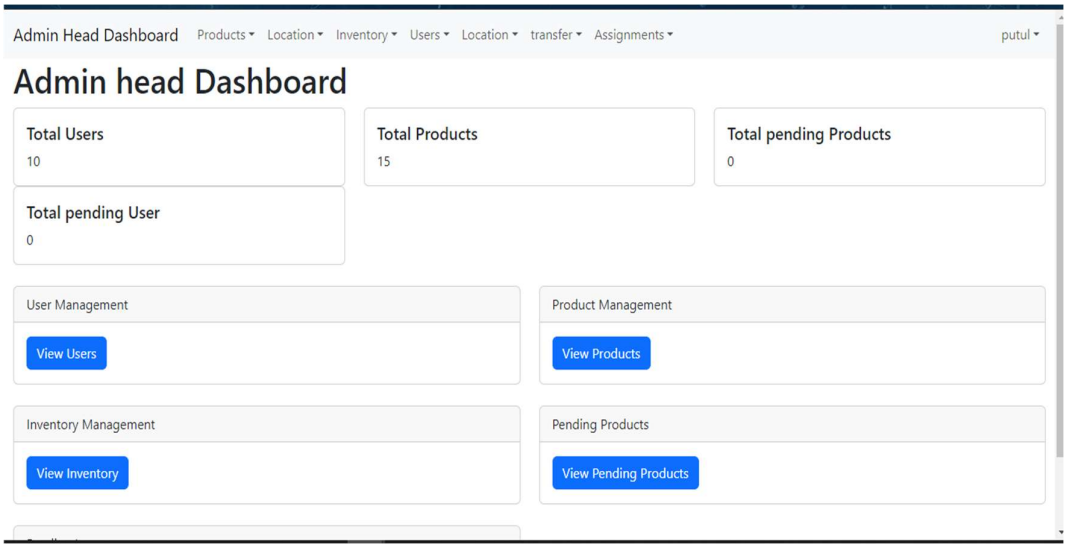


Figure 35. Admin head dashboard

Figure 35 shows an administrative dashboard within system, specifically designed for head administrators. The dashboard offers a centralized overview including the total number of users, products, pending products, pending users. Additionally, it provides access to functional areas for user management, product management, inventory management, and viewing pending products. This comprehensive dashboard serves as a valuable tool for head administrators, enabling them to efficiently manage various aspects of the system, such as approving or rejecting new user registrations and product submissions, monitoring inventory levels, and overseeing the overall operation of the application.

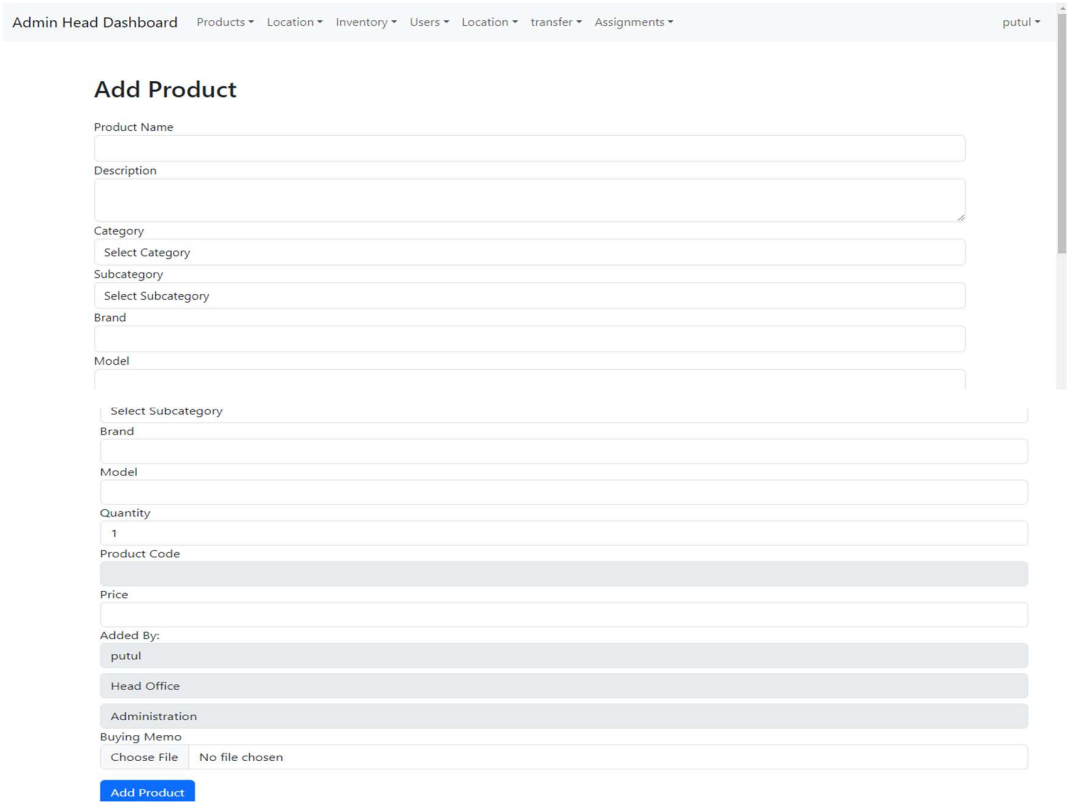


Figure 36. Add product form

Figure 36 highlights a form designed to add a new product within a specified system or application. The form requires users to input essential product details, including the product name, description, category, subcategory, brand, model, quantity, product code, price, added by user, department, and buying memo. Upon entering the necessary information and selecting the appropriate options, users can submit the form to create a new product record within the system. The "Added By" field is automatically populated with the username of the current user, and the product code is generated upon selecting the subcategory. The quantity field is set to 1 by default, and the "Buying Memo" field allows for additional notes. This form provides a structured and efficient way for administrators to add new products to the system.

Figure 37. Pending product environment

Figure 37 is a section of an administrative dashboard within system, specifically designed for head administrators. The section focuses on pending products, displaying a message indicating that there are currently no products awaiting approval or further processing. This information is useful for administrators who need to monitor the status of product submissions and ensure that all products are reviewed and approved in a timely manner.

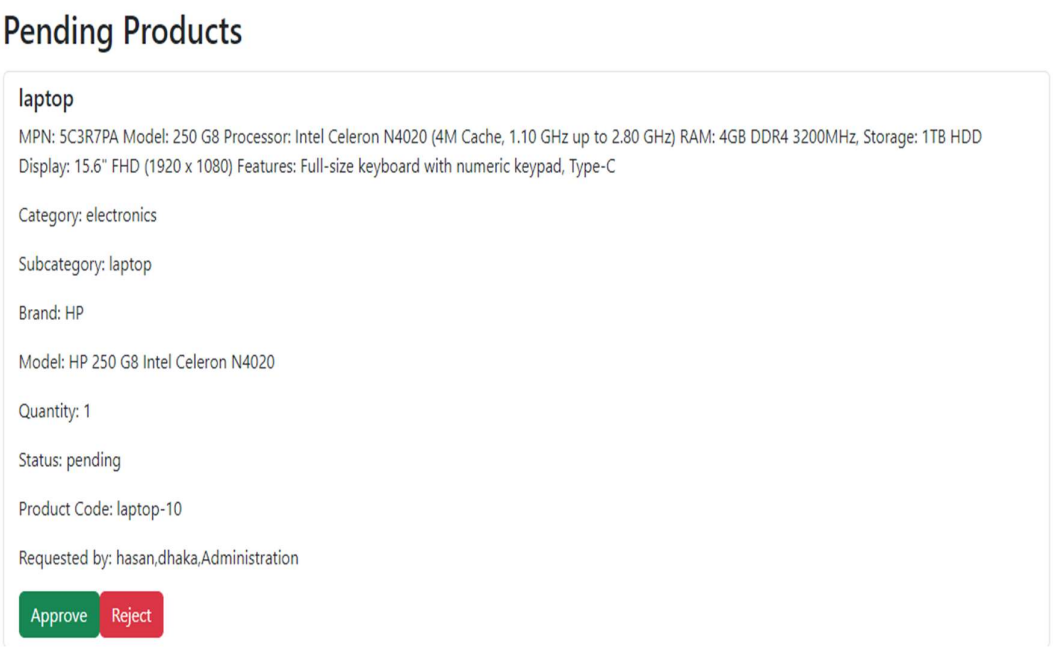


Figure 38. Pending product environment

If there is pending product, the section will display a list of these products, along with detailed information about each product, including its name, specifications,

quantity, status, and requesting user information. Additionally, the section provides buttons for approving or rejecting the products, allowing administrators to take action on pending product submissions. Upon approval, the product will be added to the inventory, and upon rejection, it will be removed from the pending list.

Figure 39. Add new location

Figure 39 shows a form to add a new location within system. The form requires to input the necessary location details, including the branch name, address, and contact information. Upon entering the required information, can submit the form to create a new location record within the system.

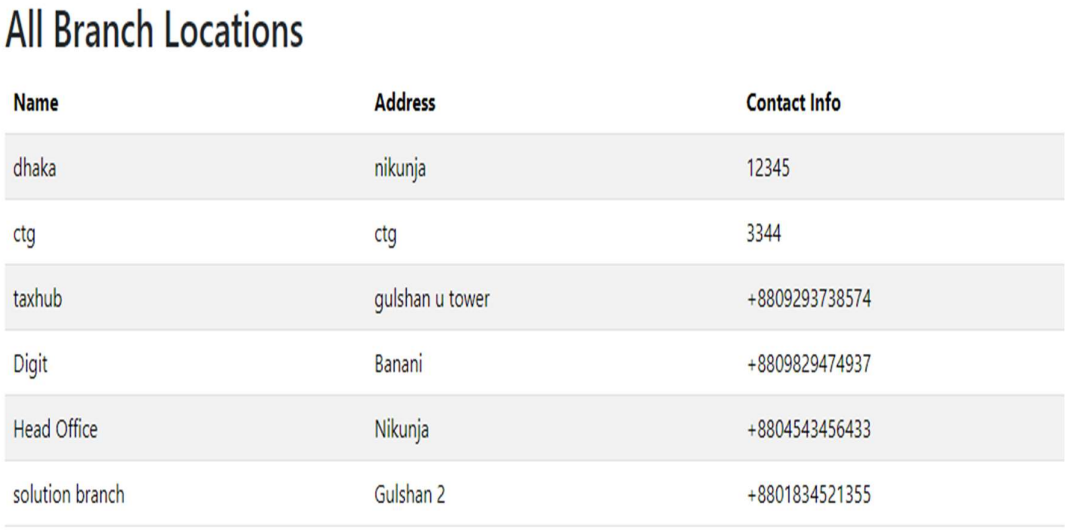


Figure 40. All branches information dashboard

Figure 40 displays a list of all branch locations within system. The list includes the name, address, and contact information for each branch.

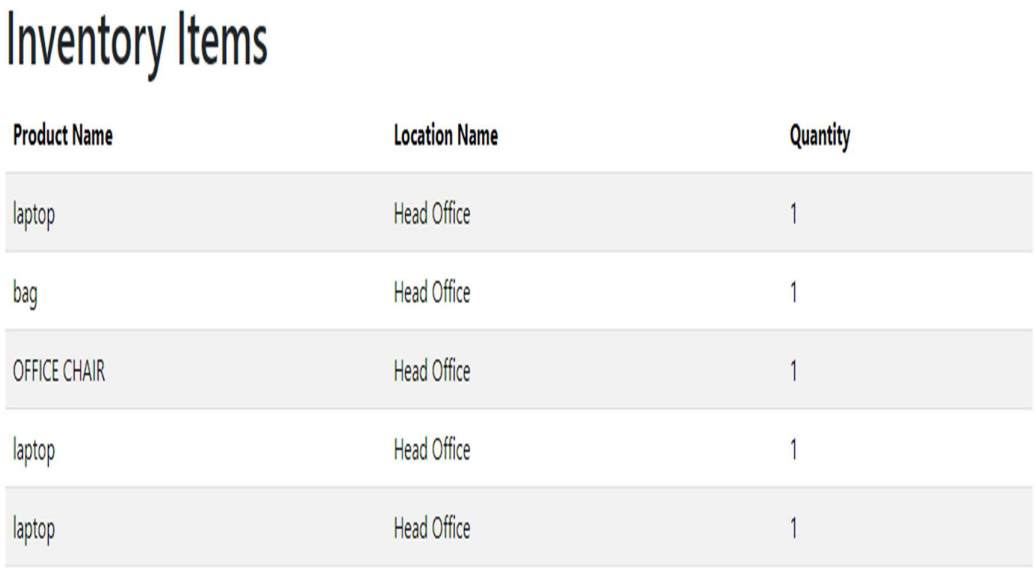


Figure 41. Available products inventory

In Figure 41, Once a product is approved by a Head Office administrator, it will be added to the branch inventory, making it visible to Branch administrators who can then manage and distribute the product within their branch. Head Office administrators are responsible for approving new products, while Branch administrators have access to a list of approved products available for distribution within their respective branches.

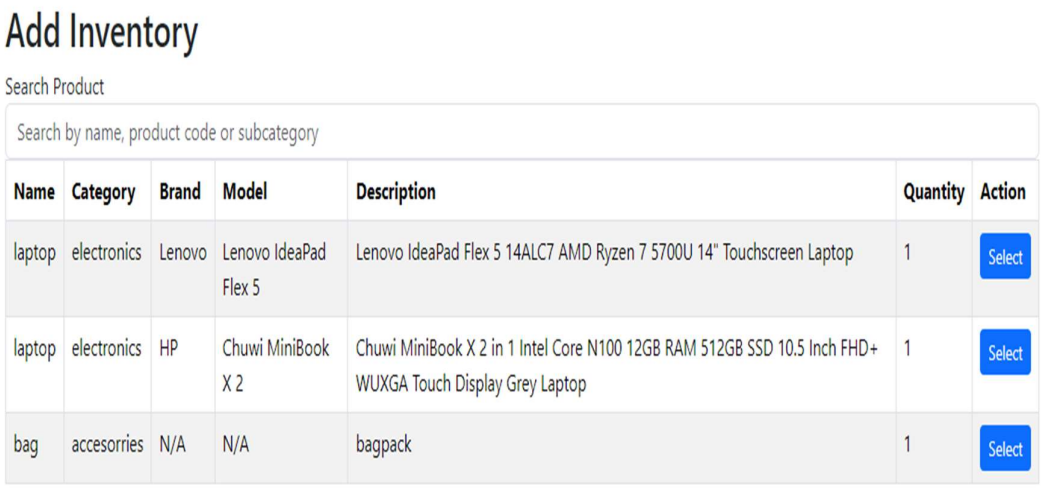


Figure 42. Available Products List and Search Function

The "Add Inventory" page allows administrators to manage product transfers from the central inventory to branch inventories. As shown in Figure 42, available products are listed, and administrators can search for specific products by name, product code, or subcategory. Once a product is located, the branch administrator can request the product for their branch, while the Head Office administrator needs to approve the request.

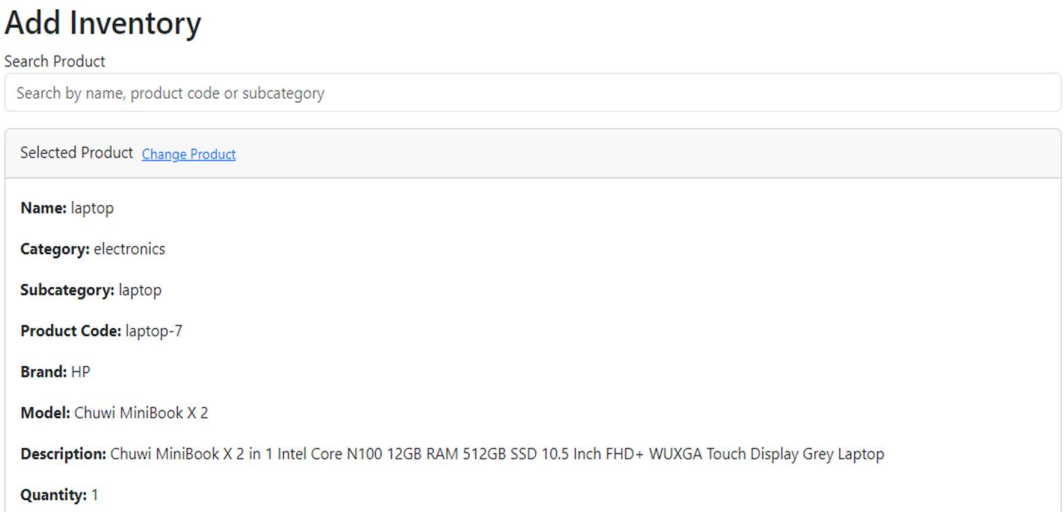


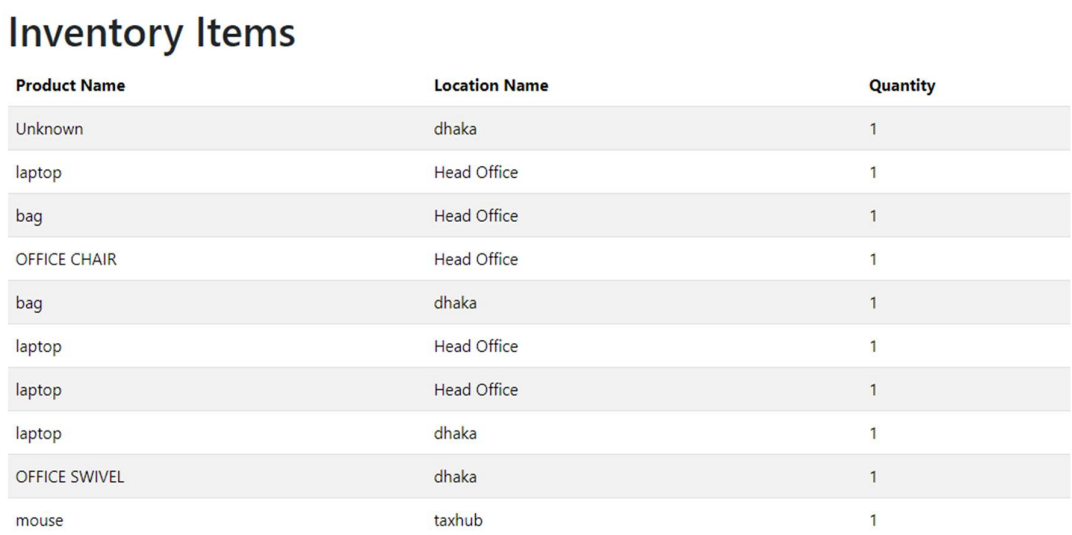
Figure 43. Product Details Display After Selection

After selecting a product, all relevant details, such as the name, category, brand, model, and description, are displayed on the page, as illustrated in Figure 43. If

the administrator needs to change the selected product, there is an option to switch to a different one by clicking "Change Product."

Figure 44. Add inventory

In Figure 44, additional fields are shown where the administrator can specify the location (branch or Head Office) and quantity. The administrator's name, department, and branch are automatically populated in the form. Once the product is added to the branch inventory, its status is set to "pending" in the inventory database until further action is taken.



Inventory Items

Product Name	Location Name	Quantity
Unknown	dhaka	1
laptop	Head Office	1
bag	Head Office	1
OFFICE CHAIR	Head Office	1
bag	dhaka	1
laptop	Head Office	1
laptop	Head Office	1
laptop	dhaka	1
OFFICE SWIVEL	dhaka	1
mouse	taxhub	1

Figure 45. Inventory items dashboard for head administrator

Figure 45 provides a comprehensive overview of all inventory items within the system, including product names, locations, and quantities. Only head administrators have access to this information, allowing them to manage and track inventory. This information allows administrators to see which products are available at each location and in what quantities. For example, the dashboard shows that there are 1 laptop available at the Head Office location, 1 bag at the Head Office location, and so on.

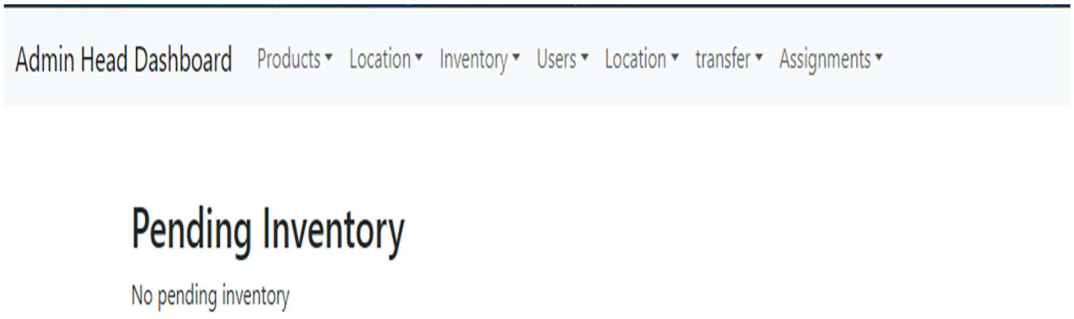


Figure 46. Admin head pending inventory dashboard

In figure 46, focuses on pending inventory, displaying a message indicating that there are currently no products awaiting approval or further processing. This information is useful for administrators who need to monitor the status of product submissions and ensure that all products are reviewed and approved in a timely manner. Pending inventory refers to products that have been added by branch administrators but require approval from head office administrators before they can be officially added to the inventory.

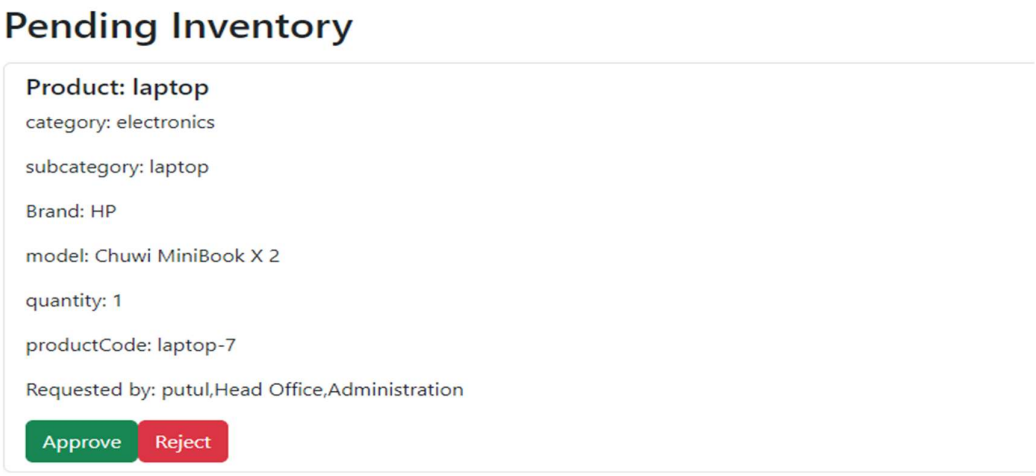
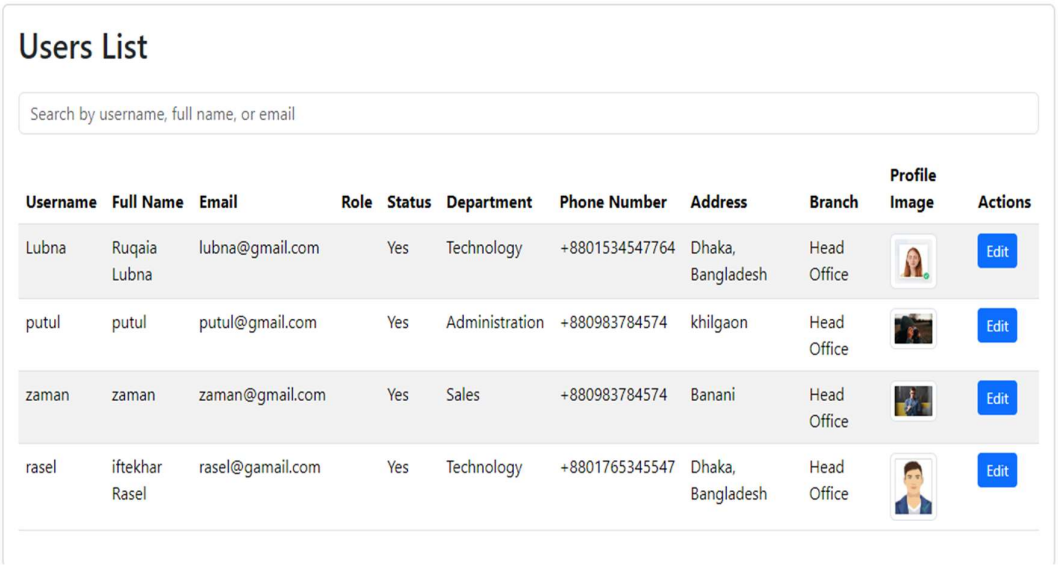


Figure 47. Admin head pending inventory dashboard

Figure 47 focuses on pending inventory, displaying details of a specific product that is currently awaiting approval or rejection. The product details include its name, category, subcategory, brand, model, quantity, product code, and requesting user information. The section provides buttons for approving or rejecting the product, allowing the administrator to take action on the pending inventory item.



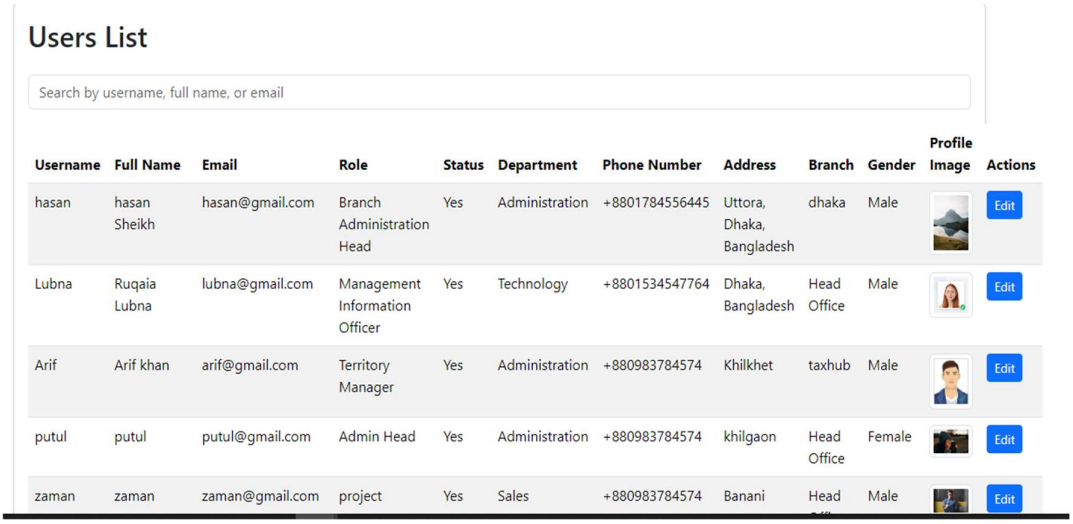
Users List

Search by username, full name, or email

Username	Full Name	Email	Role	Status	Department	Phone Number	Address	Branch	Profile Image	Actions
Lubna	Ruqaia Lubna	lubna@gmail.com		Yes	Technology	+8801534547764	Dhaka, Bangladesh	Head Office		Edit
putul	putul	putul@gmail.com		Yes	Administration	+880983784574	khilgaon	Head Office		Edit
zaman	zaman	zaman@gmail.com		Yes	Sales	+880983784574	Banani	Head Office		Edit
rasel	iftekhar Rasel	rasel@gamail.com		Yes	Technology	+8801765345547	Dhaka, Bangladesh	Head Office		Edit

Figure 48. User list dashboard

Figure 48 shows the page where administrators can view all employees within the branch and manage their details. The list includes information such as the user's username, full name, email, role, status, department, phone number, address, branch affiliation, and profile image. Administrators can update the user's status to either "approved" or "not approved," enabling control over user access and permissions within the system. The data presented in the image is fictional and does not represent real users or information.



Users List

Search by username, full name, or email

Username	Full Name	Email	Role	Status	Department	Phone Number	Address	Branch	Gender	Profile Image	Actions
hasan	hasan Sheikh	hasan@gmail.com	Branch Administration Head	Yes	Administration	+8801784556445	Uttora, Dhaka, Bangladesh	dhaka	Male		Edit
Lubna	Ruqaia Lubna	lubna@gmail.com	Management Information Officer	Yes	Technology	+8801534547764	Dhaka, Bangladesh	Head Office	Male		Edit
Arif	Arif khan	arif@gmail.com	Territory Manager	Yes	Administration	+880983784574	Khilkhet	taxhub	Male		Edit
putul	putul	putul@gmail.com	Admin Head	Yes	Administration	+880983784574	khilgaon	Head Office	Female		Edit
zaman	zaman	zaman@gmail.com	project	Yes	Sales	+880983784574	Banani	Head	Male		Edit

Figure 49. User/Employee list dashboard

In Figure 49, this allows administrators to view and manage user information across all branches. The search bar enables administrators to quickly locate specific users based on their username, full name, or email. Administrators can update the user's status to either approved or not approved. This feature allows administrators to manage user access and permissions within the system. The information displayed in the image provides a comprehensive overview of the users within the system, including their roles, departments, and branch affiliations. The data presented in the image is fictional and does not represent real users or information.

Figure 50. Handover of product to another employee

Figure 50 presents a form designed to initiate a product transfer or handover between employees. The form requires users to select the transferring user, receiving user, transferring branch, receiving branch, transfer date, requested by user, approved by user, approval date, and status. Upon entering the necessary information, users can submit the form to create a new transfer record within the system, initiating the process of transferring the specified product from one employee to another.

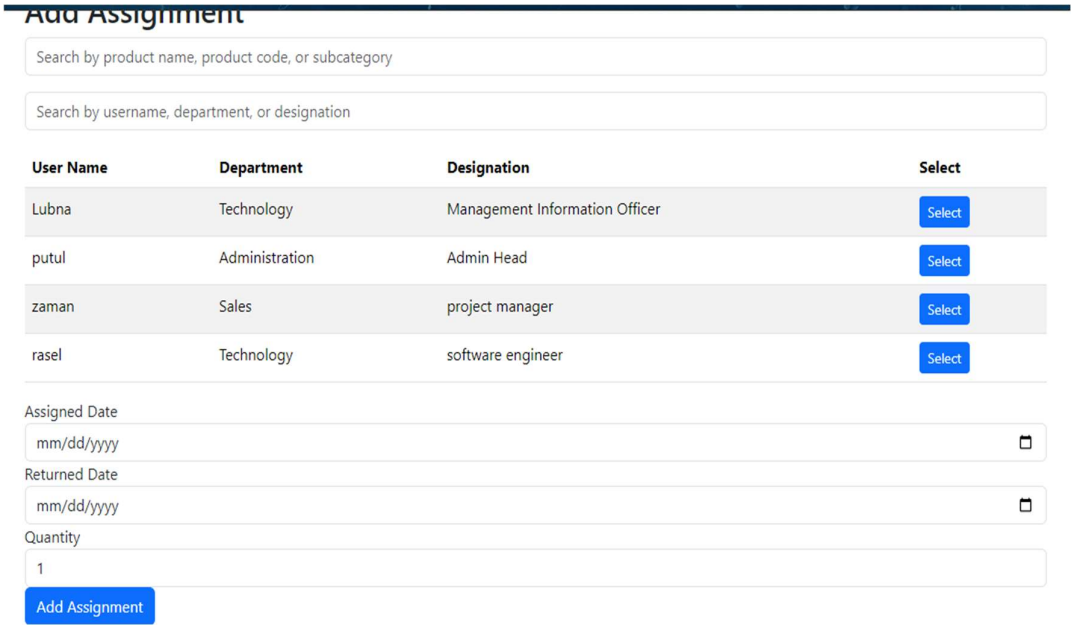


Figure 51. Assigning of product to employee

Figure 51 presents a form designed for administrators to assign a specific product to an employee. The form includes search fields for filtering users by product name, product code, or subcategory, and searching for employees by username, department, or designation. Administrators can select the transferring user, receiving user, assigned date, expected returned date, and quantity. Upon entering the necessary information, administrators can submit the form to create a new assignment record within the system, associating the specified product with the designated employee.

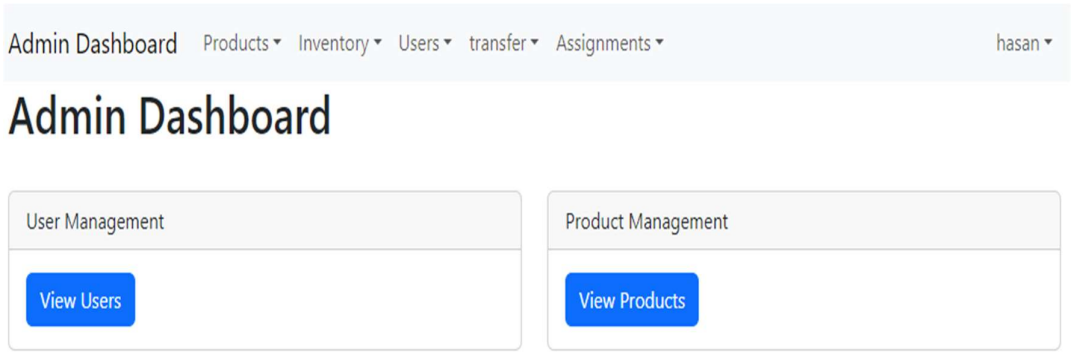


Figure 52. Admin dashboard

Figure 52 shows an administrative dashboard, specifically designed for branch administrators. The dashboard offers a centralized overview of key functionalities related to user management and product management. Branch administrators can use this dashboard to view and manage user information and product details within their respective branches.

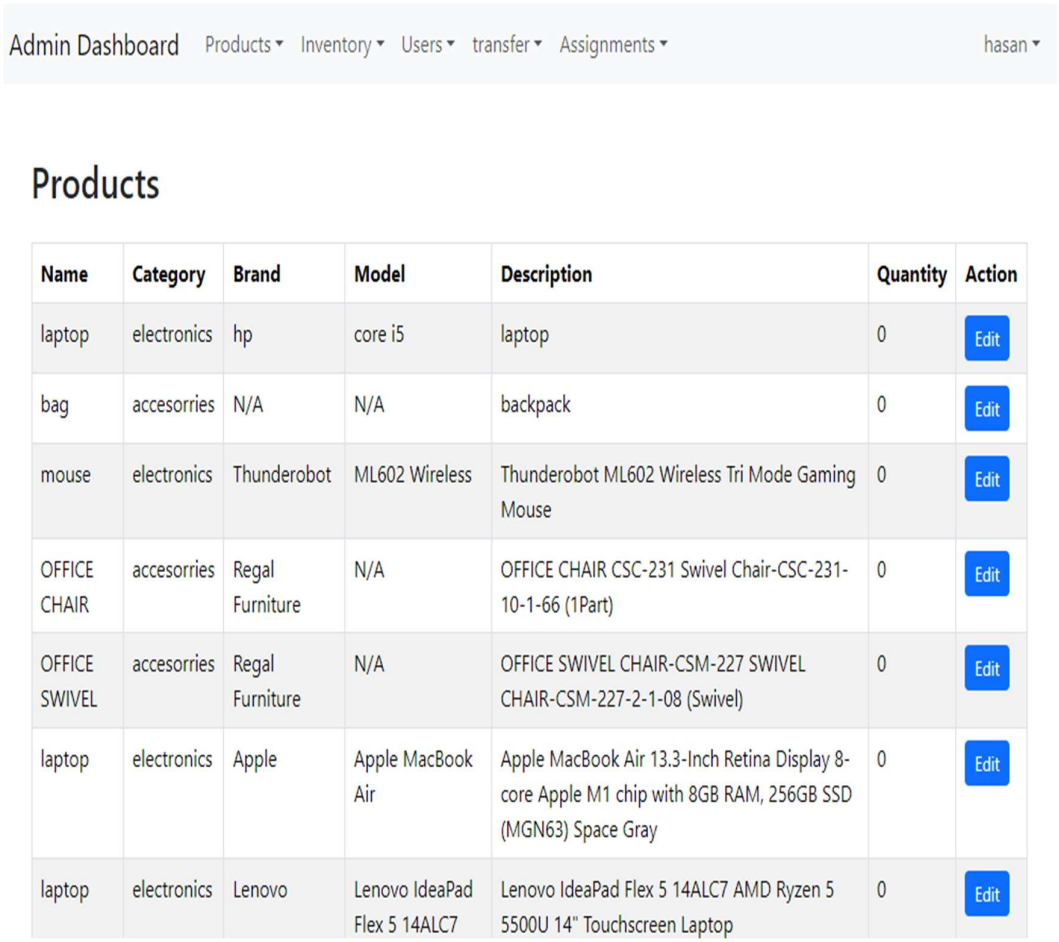


Figure 53. Admin dashboard to monitor branch inventory

Figure 53 shows a section of an administrative, specifically designed for branch administrators. The section focuses on inventory items, displaying a list of products that are currently available within the branch. The list includes details about each product, such as its name, category, brand, model, description, and quantity. This information provides a comprehensive overview of the inventory within the branch, allowing administrators to track and manage the availability of products.

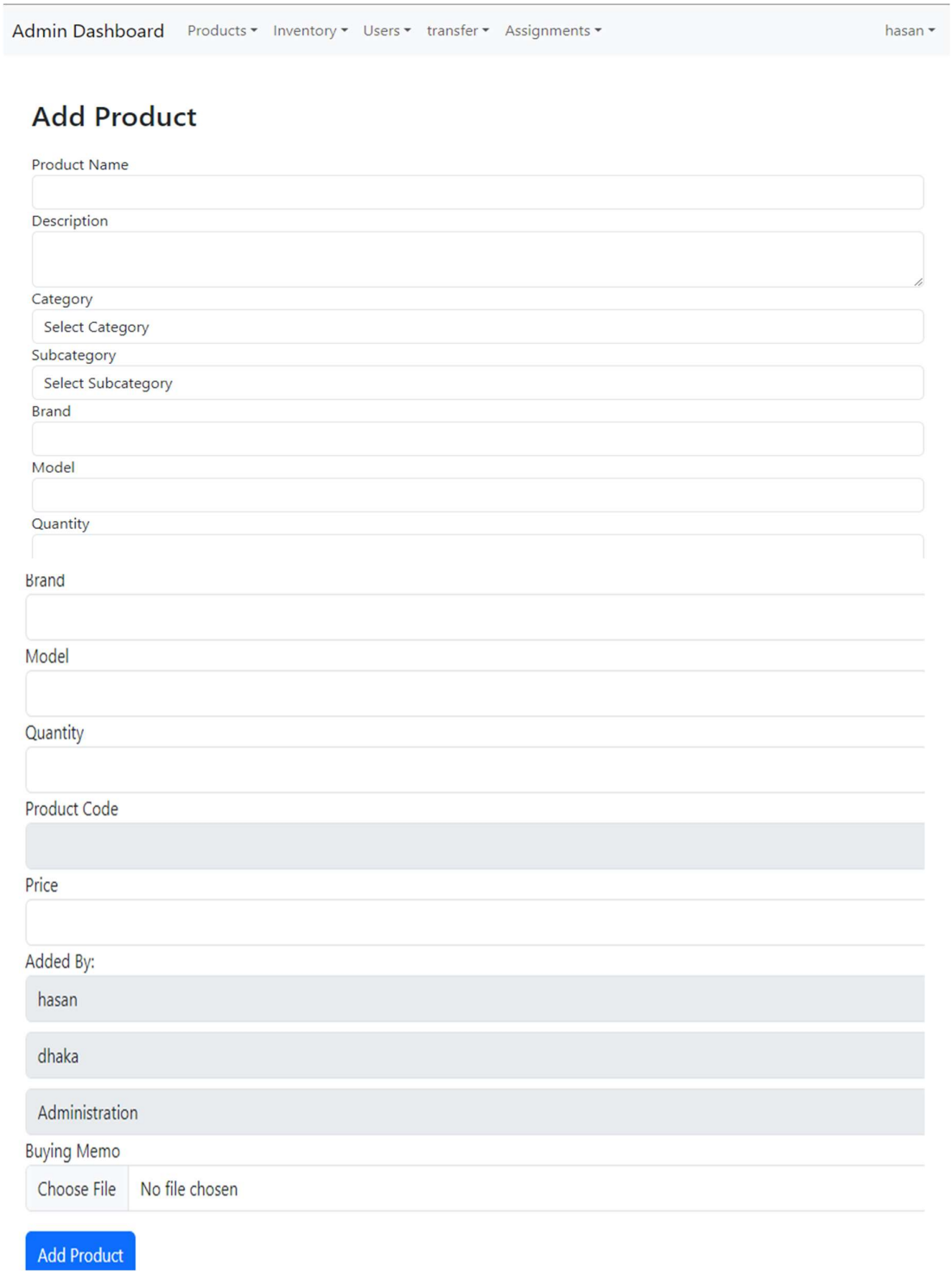


Figure 54: Add product form

Figure 54 shows a form designed to add a new product. The form requires users to input essential product details, including the product name, description, category, subcategory, brand, model, quantity, product code, price, added by user,

department, and buying memo. Upon entering the necessary information and selecting the appropriate options, users can submit the form to create a new product record within the system. Notably, the "Added By" field is automatically populated with the username of the current user, and the product code is generated upon selecting the subcategory. The quantity field is set to 1 by default, allowing for easy adjustments if needed. This form provides a structured and efficient way for administrators to add new products to the system.

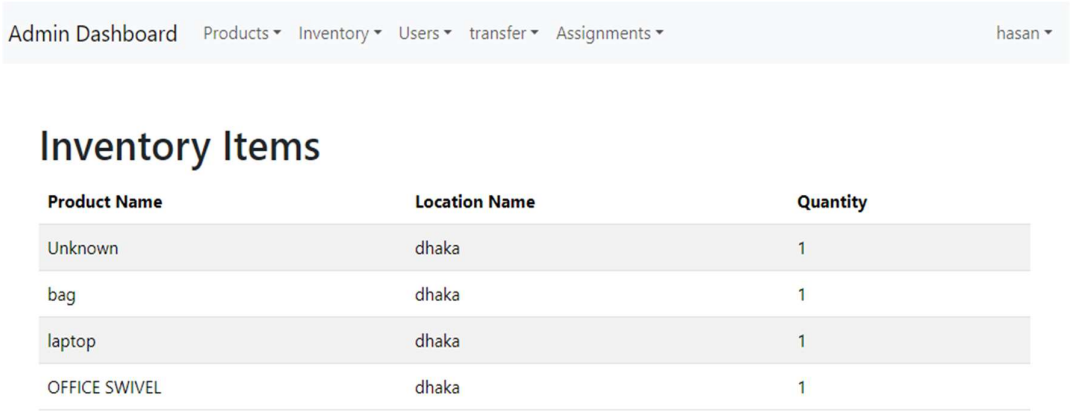


Figure 55. Available inventory items

Figure 55 shows a section of an administrative, specifically designed for branch administrators. The section focuses on inventory items, displaying a list of products that are currently available within the branch. The list includes details about each product, such as its name, location, and quantity. This information provides a comprehensive overview of the inventory within the branch, allowing administrators to track and manage the availability of products.

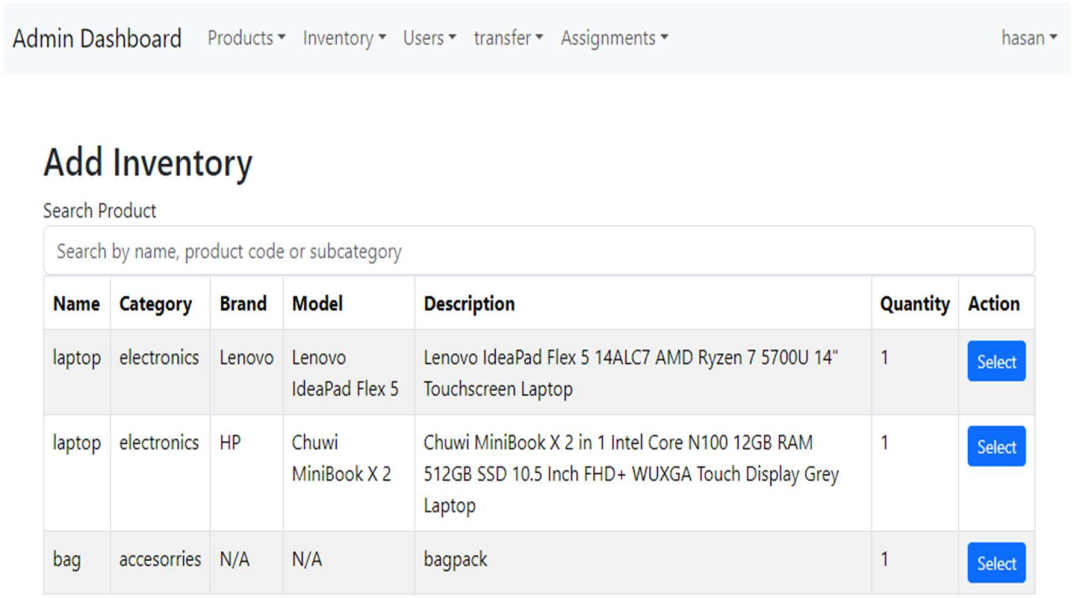


Figure 56. Add Inventory

Figure 56 shows the add inventory page from the Admin Dashboard, displaying a table of products available for selection. The table includes columns for Name, Category, Brand, Model, Description, Quantity, and an Action column with a "Select" button for each product. A search bar is available at the top, allowing users to search for products by name, product code, or subcategory. This page is used to move existing products into a specific branch's inventory, rather than adding new products to the overall inventory.

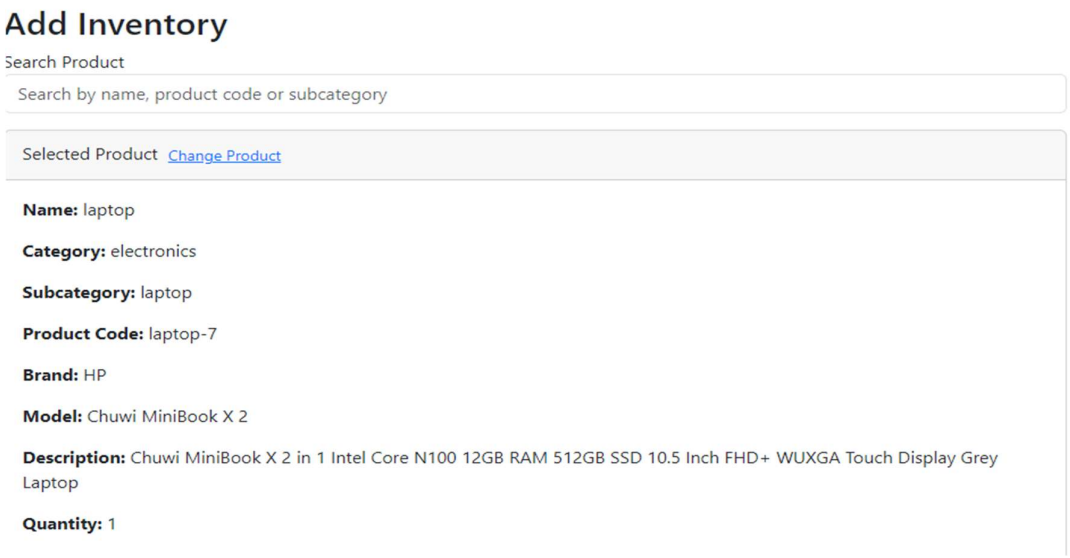


Figure 57. Product details

Image 57 shows the Add Inventory page after a product has been selected. The selected product is a laptop, and its detailed information is displayed, including Name, Category, Subcategory, Product Code, Brand, Model, and a full description of the product's specifications. The user has the option to change the selected product by clicking "Change Product." The page still includes the search bar at the top for finding additional products.

Figure 58. Branch & department information

Image 58 shows the Add Inventory page after the product selection, now including a form to add inventory details. The form fields include Location, Quantity, and Added By. The previously selected product's information—such as Name, Category, Subcategory, Product Code, Brand, Model, Description, and Quantity—is displayed above the form. The user has entered "dhaka" as the location and "hasan" as the person adding the inventory. An "Add Inventory" button is visible at the bottom of the form, ready to submit the inventory details.

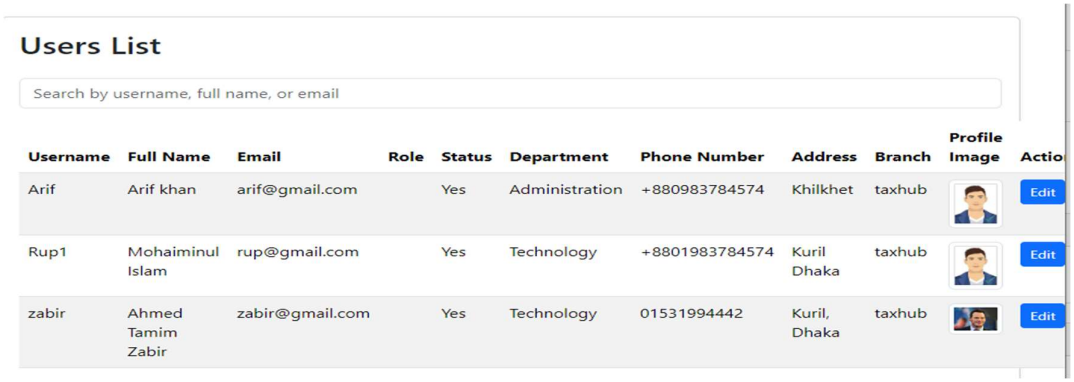


Figure 59. User/Employee list

Figure 59 shows a user list within an administrative dashboard, allowing administrators to view and manage user information. The list includes details such as the user's username, full name, email, role, status, department, phone number, address, branch affiliation, profile image, and actions. The search bar enables administrators to quickly locate specific users based on their username, full name, or email. Additionally, administrators can update the user's status to either approved or not approved. This feature allows administrators to manage user access and permissions within the system. The data presented in the image is fictional and does not represent real users or information.

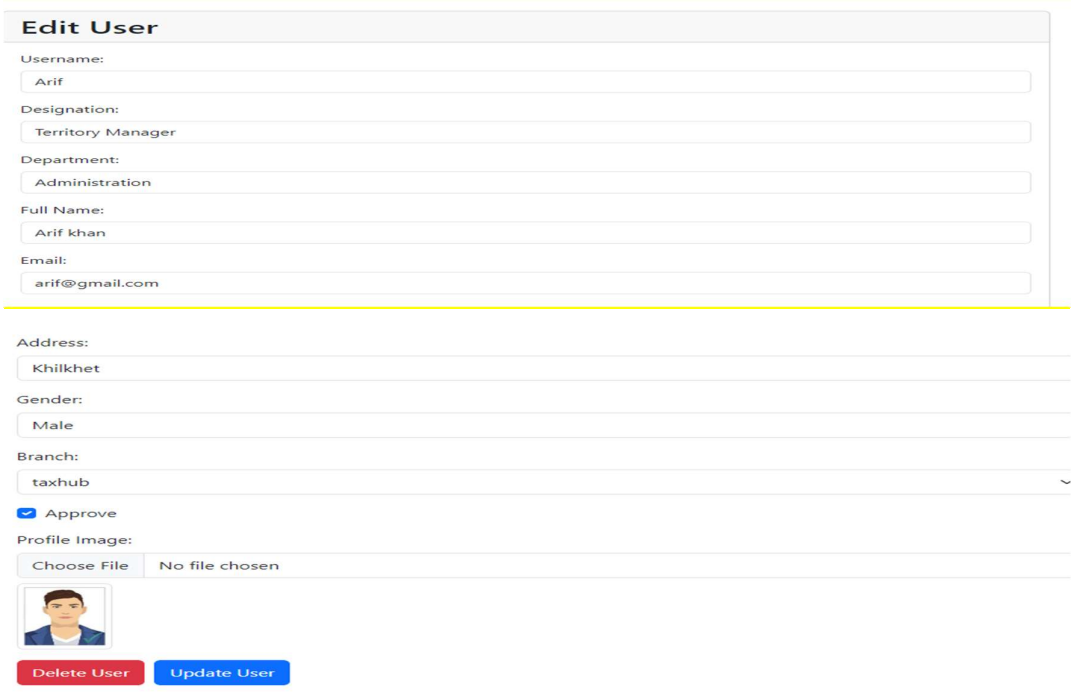


Figure 60. User status dashboard

Figure 60 displays a user management form within an administrative dashboard, allowing administrators to view, edit, and delete user information. The form includes fields for the user's username, designation, address, gender, branch affiliation, profile image, and status. Administrators can update the user's information by making changes to the relevant fields and clicking the "Update User" button. Additionally, administrators can delete a user by clicking the "Delete User" button. The "Choose File" button allows administrators to upload a new profile image for the user.

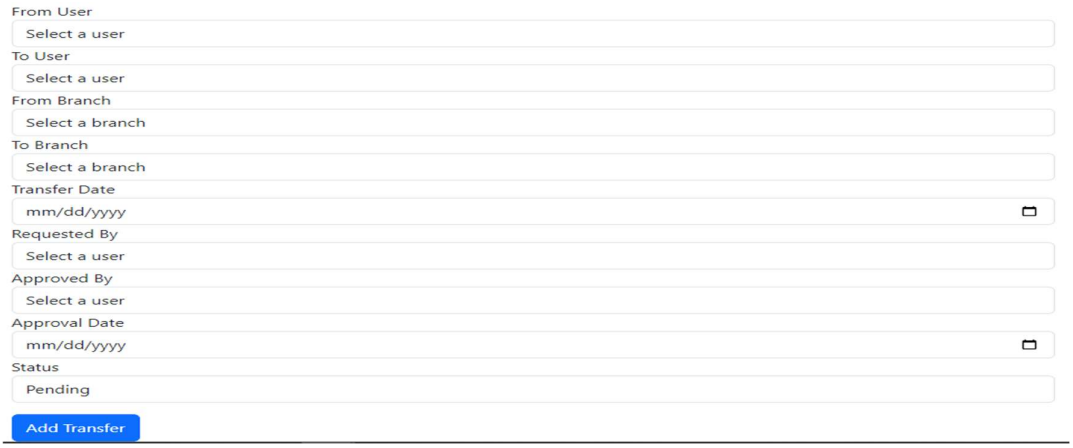


Figure 61. Transfer or handover of product

Figure 61 shows a form designed to initiate a product transfer or handover between employees. The form requires users to select the transferring user, receiving user, transferring branch, receiving branch, transfer date, requested by user, approved by user, approval date, and status. Upon entering the necessary information, users can submit the form to create a new transfer record within the system, initiating the process of transferring the specified product from one employee to another.

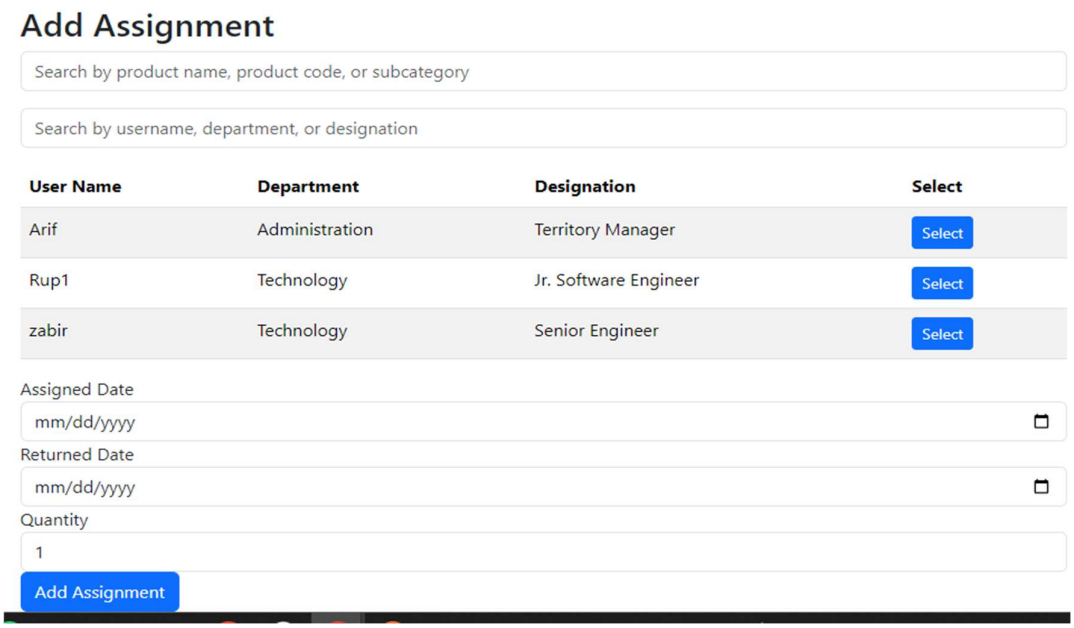


Figure 62. Assigning product to employee

Figure 62 shows a form designed to assign a specific product to an employee. The form includes search fields for filtering users by product name, product code, or subcategory, and searching for employees by username, department, or designation. Users can select the desired product and employee from the respective lists, enter the assigned date and expected returned date and specify the quantity. Upon entering the necessary information, users can submit the form to create a new assignment record within the system, associating the specified product with the designated employee.

4.7.3 A Look Inside the Inventory Database: Navigating with MongoDB Compass

In this project, MongoDB was chosen as the database due to its flexibility, scalability, and ability to handle complex and unstructured data. As a NoSQL database, MongoDB uses a document-oriented format that allows for dynamic and evolving data models, making it ideal for modern web applications. Its schema-less nature offers an advantage by allowing changes to the data structure without the need for major modifications, which is often required in traditional relational databases. MongoDB's scalability makes it suitable for handling large datasets and expanding as the project grows. This allows the system to manage increasing amounts

of data efficiently without compromising performance. Below is an overview of the database, highlighting the collections and their key fields with images for better understanding.

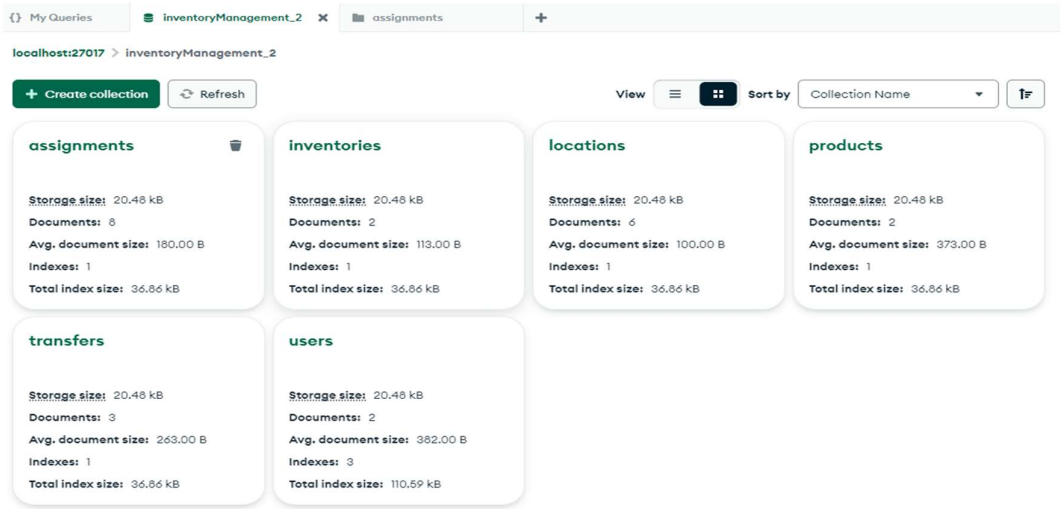


Figure 63: Overview of MongoDB Collections

Figure 63 displays a MongoDB database interface for project named inventory-Management_2, featuring six collections: assignments, inventories, locations, products, transfers, and users. Each collection's summary includes details such as the storage size, number of documents, average document size. The snapshot provides an organized overview of the database structure, understanding data within the project.

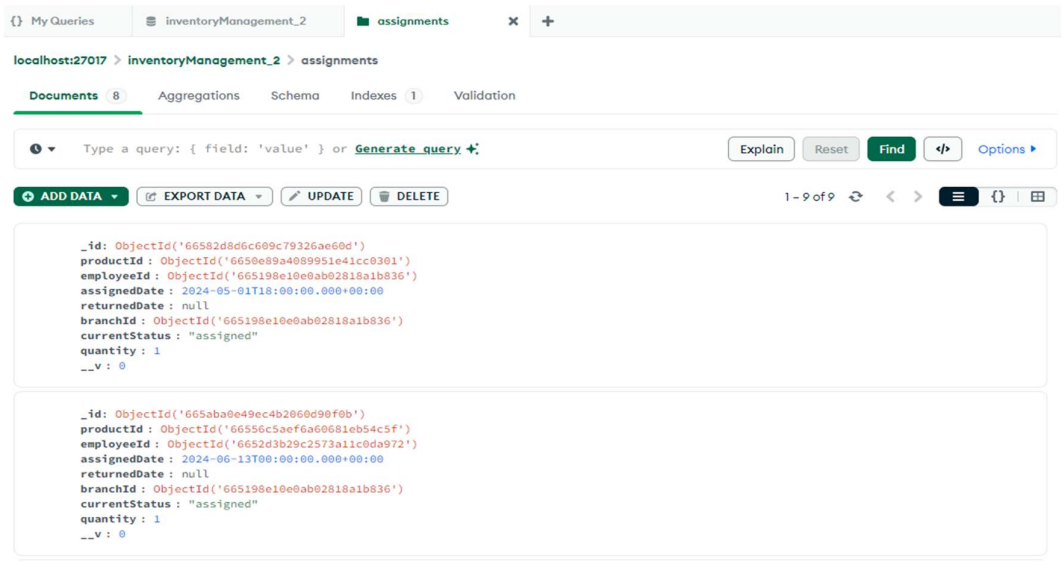


Figure 64: Collection fields name and documents store dashboard

Figure 64 presents a interface designed for managing MongoDB database. Specifically, it focuses on the "assignments" collection within this database. Each document within the "assignments" collection is structured with several fields: a unique identifier (id), a reference to the associated product (productId), a reference to the employee responsible (employeeId), the date the assignment was made (assignedDate), the date it was returned (returnedDate if applicable), a reference to the branch where it was made (branchId), the current status of the assignment (currentStatus, either "assigned" or "returned"), and the quantity of the product involved (quantity). The user interface provides tools for interacting with the "assignments" collection, including export existing data, update existing documents, and delete unwanted documents.

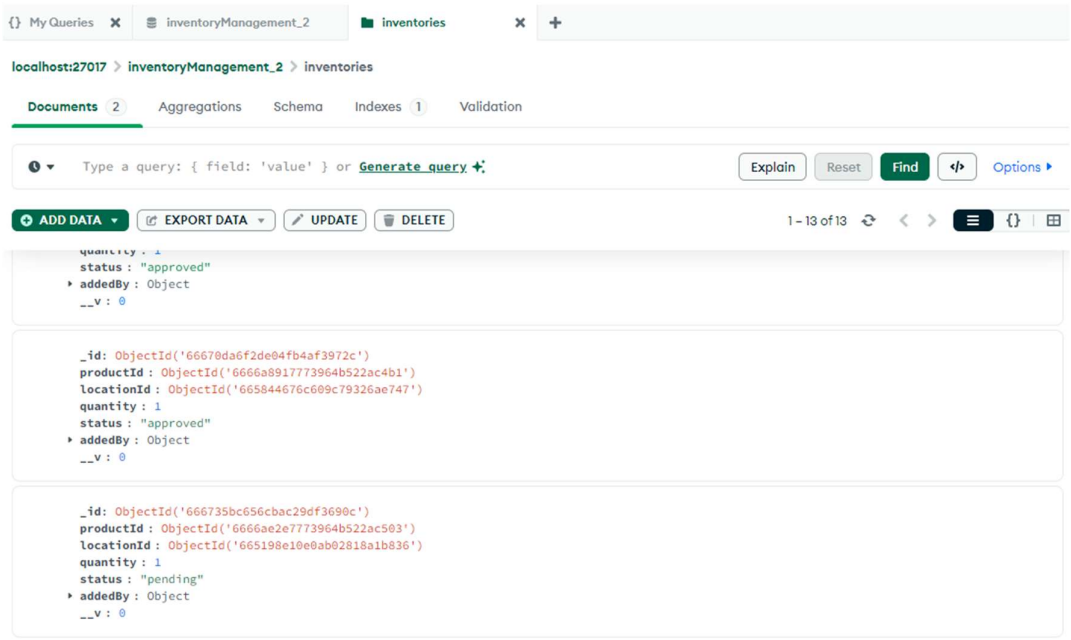


Figure 65. Store dashboard for inventory documents

Figure 65 depicts a interface for managing a MongoDB database. Specifically, it focuses on the "inventories" collection within this database. Each document within the "inventories" collection is structured with several fields: a unique identifier (id), a reference to the associated product (productId), a reference to the location where it's stored (locationId), the quantity of the product in the inventory (quantity), the current status of the inventory (status, either "approved" or "pending"), and an object contains additional information about the user who added the product (addedBy). The user interface provides tools for interacting with the "inventories" collection, including the ability to export existing data, update existing documents, and delete unwanted documents.

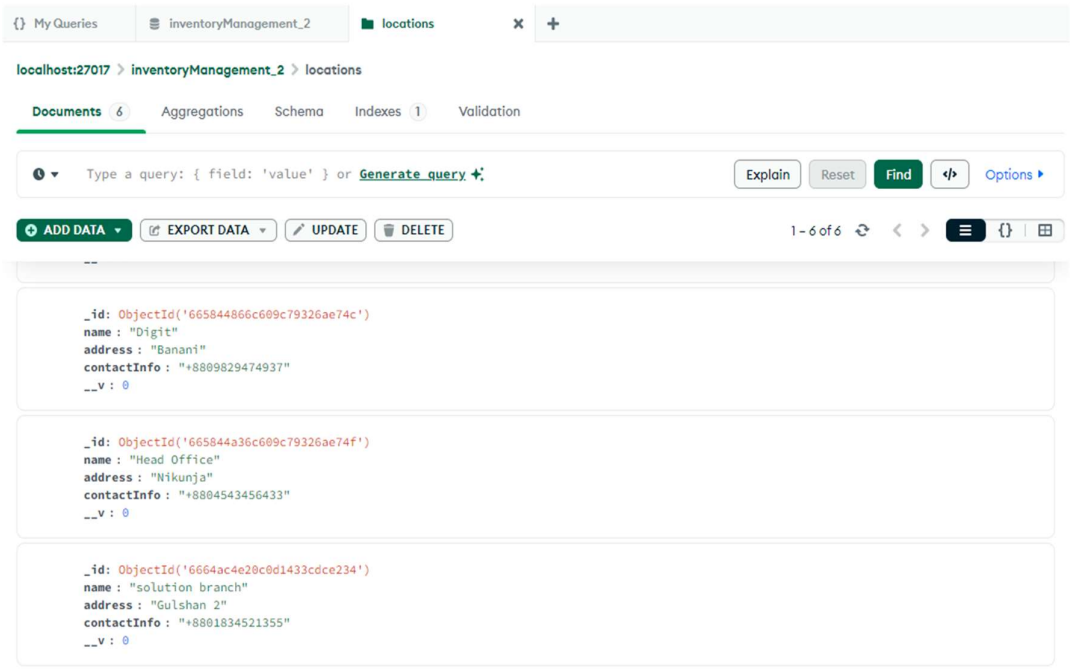


Figure 66. Store dashboard for location information

Figure 66 focuses on the "locations" collection within this database. Each document within the "locations" collection is structured with several fields: a unique identifier (id), the name of the location (name), the address of the location (address), and the contact information for the location (contactInfo).

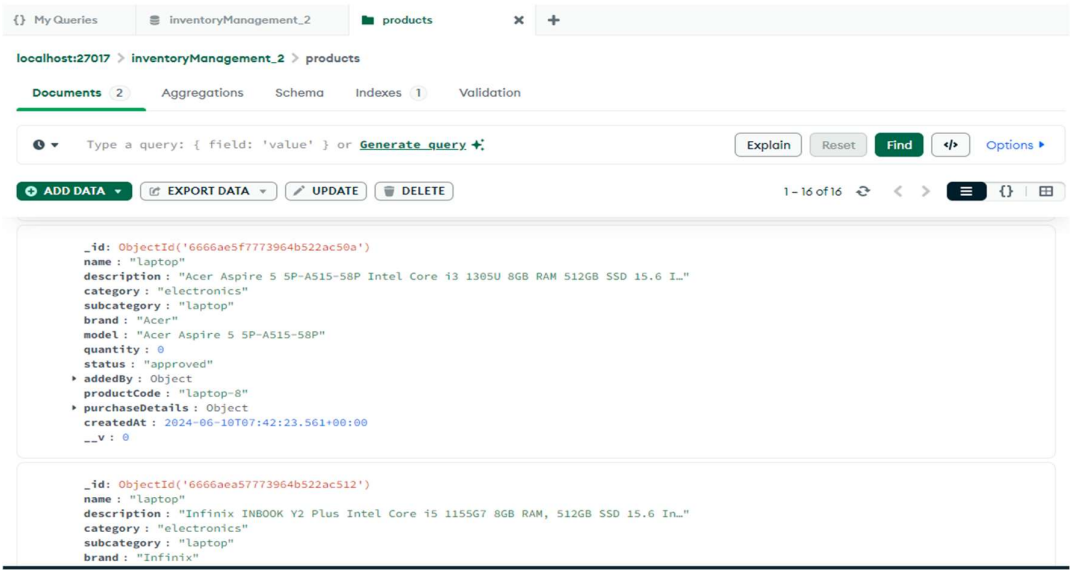


Figure 67. Store dashboard for product documents

Figure 67 focuses on the "products" collection within this database. Each document within the "products" collection is structured with several fields: a unique identifier (id), the name of the product (name), a description of the product (description), the product category (category), the product subcategory (subcategory), the brand of the product (brand), the model of the product (model), the quantity of the product in stock (quantity), the current status of the product (status, either "approved" or "pending"), an object containing additional information about the user who added the product (addedBy), the product code (productCode), an object containing purchase details (purchaseDetails), and the date and time the product was created (createdAt).

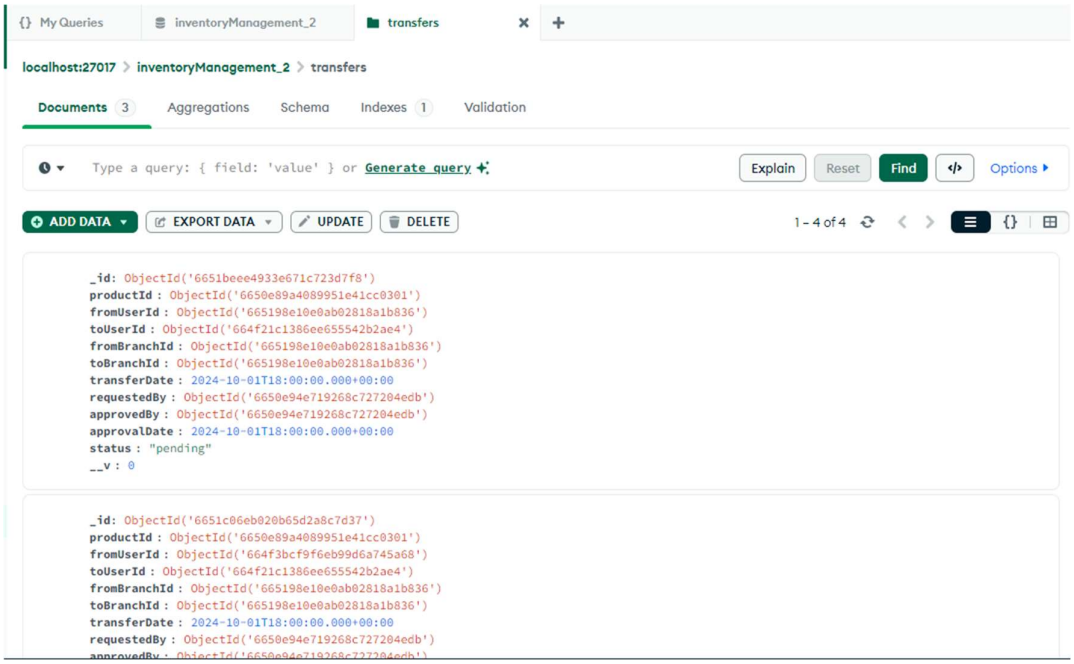


Figure 68. Store dashboard for transfer documents

Figure 68 represents the fields of the transfer information. In the documents, transfer information will be stored whenever a transfer of inventories occurs.

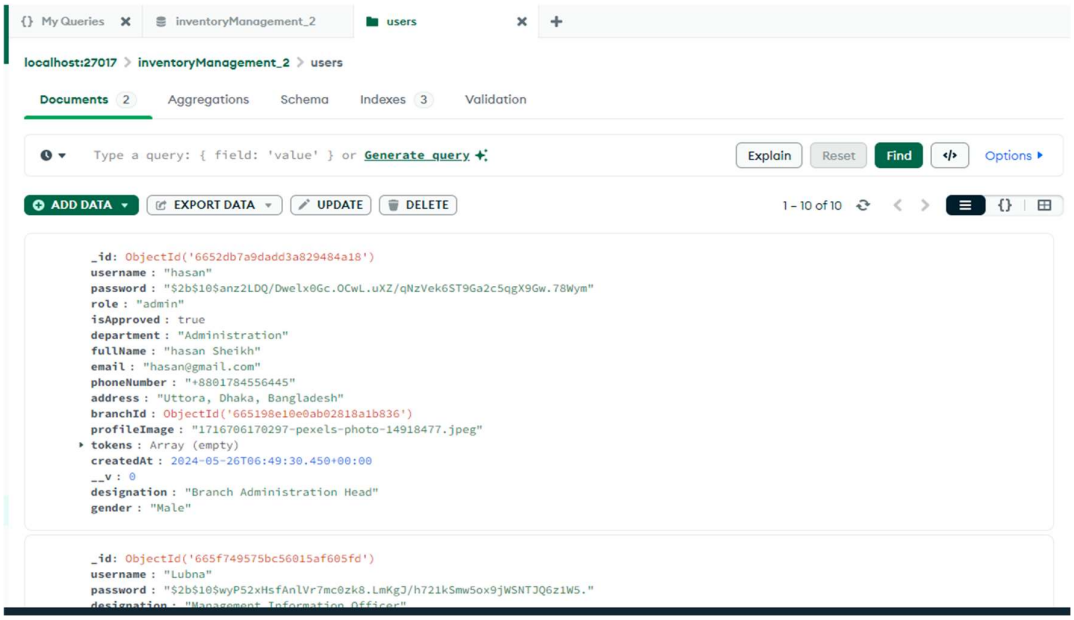


Figure 69: Store dashboard for user information

Figure 69 focuses on the "transfers" collection within this database. Each document within the "transfers" collection is structured with several fields: a unique identifier (id), a reference to the associated product (productId), a reference to the user who initiated the transfer (fromUserId), a reference to the user who received the transfer (toUserId), a reference to the branch where the transfer originated (fromBranchId), a reference to the branch where the transfer was received (toBranchId), the date and time the transfer was requested (transferDate), the user who requested the transfer (requestedBy), the user who approved the transfer (approvedBy), the date and time the transfer was approved (approvalDate), and the current status of the transfer (status, either "pending" or "approved").

5 DISCUSSION

In the continuously evolving technology, efficient inventory management plays a vital role for businesses of all sizes. Inventory management plays an essential role in the working efficiency, productivity and profitability of businesses across various industries. Traditional methods of inventory management often lead to inefficiencies and inaccuracies. Uses of contemporary frameworks and tools can expressively enhance the developing of management system process. The proposed project planned to utilize the MERN stack to implement an inventory management system followed by API testing using Postman.

MongoDB, a NoSQL database, offers flexible and well-suited handling of diver's inventory data. Express.js, a simple web application framework for Node.js which shorten the method of building server-side application and APIs that helps an uninterruptable communication between the frontend and backend components. React.js is a Java script library for executing user interfaces and uses a simulated React that allows the manufacture of co-operating user interfaces for inventory management software. Node.js is a runtime environment for performing Java script code server-side and for performing high performance of backend logic and data processing.

Operative API testing is crucial for verifying the functionality, performance of software, and its consistency and its consistency. Postman is a widespread API testing tool for designing, debugging and testing of APIs. By methodically testing cases and situations, developers can highly rely on that every single API endpoint will eventually act predictably and meet the stated requirements.

All projects have some scopes to make improvement in the future. Corresponding project also has some future development which will make it more resourceful and more operative. The proposed project also has several future scopes and enhancement. Below some of the future development has been provided:

- Real time inventory tracking, predicting by analysis and incorporation with third-party services
- Addition of instinctual interfaces and accessibility features

- Unremitting optimization of performance and security

In conclusion, having an enhanced inventory management solution helps organizations that deal with inventories to unlock new opportunities and better performance, thereby the management solution lingers ahead in a tough and competitive business atmosphere.

REFERENCES

An Introduction to Mongoose for MongoDB and Node.js", https://code.tutsplus.com/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527a

A Review Paper on Bootstrap Framework - SURAJ SHAHU GAIKWAD1 , PROF PRATIBHA ADKAR. APR 2019 | IRE Journals | Volume 2 Issue 10 | ISSN: 2456-8880

Behavior of MVC (Model View Controller) based Web Application developed in PHP and .NET framework - Manisha Jailia, Ashok Kumar, Manisha Agarwal & Isha Sinha. [IEEE 2016 International Conference on ICT in Business Industry & Government (ICTBIG) - Indore, India (2016.11.18-2016.11.19)]

Designing an MVC Model for Rapid Web Application Development
Dragos - Paul Pop & Adam Nelu Altar Samuel. Procedia Engineering 69 (2014) 1172–179

Development of a Self-Diagnostic System Integrated into a Cyber-Physical System - Domingos F. Oliveira, João P. Gomes , Ricardo B. Pereira , Miguel A. Brito and Ricardo J. Machado. August 2022 Computers 11(9):131

Enhancing User Experience in Content Management through MERN Stack Integration - Karamjeet Kaur , Gaurav Rai , Lata Phartyal , Shivani Rautela , Himanshu. ISSN : 0044-0477. VOLUME 23 : ISSUE 02 (Feb) – 2024

INVENTORY MANAGEMENT INTRODUCING A FRAMEWORK TO ASSESS OPERATIONAL PERFORMANCE,
https://filelist.tudelft.nl/TBM/Over%20faculteit/Afdelingen/Engineering%20Systems%20and%20Services/People/Professors%20emeriti/Jan%20van%20den%20Berg/MasterPhdThesis/GvHeck_THESIS_FINAL.pdf

Inventory Management System - Rishabh gupta, Ashish, Aman yadav. Vol-ume 10, Issue 4 April 2022 | ISSN: 2320-2882

International Journal of Research Publication and Reviews, Vol 4, no 5, pp 342-345 May 2023. Inventory Management Platform Using MERN Stack Ap-plication - Dr. M. Sujithra , Hanish S , Akschaya B , Danvanth S , Sivasakthi G

Introduction to MERN Stack & Comparison with Previous Technologies - Prof. Yogesh Kadam, Akhil Goplani, Shubit Mattoo, Shashank Kumar Gupta, Dar-shan Amrutkar, Prof. Dr. Jyoti Dhanke .June 2023 European Chemical Bulle-tin 12(Special Issue 4):14382-14386.

Inventory Management Platform Using MERN Stack Application - Dr. M. Su-jithra, Hanish S, Akschaya B, Danvanth S, Sivasakthi G. International Journal of Research Publication and Reviews, Vol 4, no 5, pp 342-345 May 2023

Inventory Management Platform Using MERN Stack Application - Dr. M. Su-jithra, Hanish S, Akschaya B, Danvanth S, Sivasakthi G. International Journal of Research Publication and Reviews, Vol 4, no 5, pp 342-345 May 2023

Journal of Applied Technology and Innovation (e -ISSN: 2600-7304) vol. 7, no. 3, (2023). Inventory Management Systems (IMS) - Chan Chin Wei, Sa-thiapriya A/P Ramia and Nurul Farhaini Razali, Asia Pacific University of Technology and Innovation (APU).

MVC Architecture: A Detailed Insight to the Modern Web Applications Devel-opment - Abdul Majeed and Ibtisam Rauf, Korea Aerospace University, South Korea. Volume 1 - Issue 1, September 26, 2018.

Performance Optimization using MERN stack on Web Application -Sourabh Mahadev Malewade & Archana Ekbote. International Journal of Engineering Research & Technology. ISSN: 2278-0181. Vol. 10 Issue 06, June-2021

Performance Optimization using MERN stack on Web Application -Sourabh Mahadev Malewade & Archana Ekbote. International Journal of Engineering Research & Technology. ISSN: 2278-0181. Vol. 10 Issue 06, June-2021

THE EVOLUTION OF INVENTORY MANAGEMENT,
https://www.odysseydcs.com/the-evolution-of-inventory-management/

Visual Studio Code Distilled" - Evolved Code Editing for Windows, macOS, and Linux, Second Edition. Alessandro Del Sole. 2021

APPENDICES

Appendix 1. GitHub Link

GitHub: https://github.com/Iftekhe/Inventory-management-system