**Inventory Management System**

**Deliverable -3**

**Code Crafters**

**Group List**

1. Sameera Ellanki
2. Varsha Jampala
3. Nikhil Kuchipudi
4. Naga Shivani Pinnamaraju
5. Jaswanth Sai Donepudi
6. Manogna Reddy Anugu
7. Atluri Varun Chowdary
8. Naga Bindu Kondaveeti

Index

[1. Requirements specification for phase one 1](#_Toc161674598)

[1.1 Functional Requirements covered under phase one. 1](#_Toc161674599)

[1.1.1 REQ-1 Inventory Management Requirements 1](#_Toc161674600)

[1.1.2 REQ-4 Supplier and Customer Requirements 1](#_Toc161674601)

[1.1.3 REQ-7 User Interface Requirements 2](#_Toc161674602)

[1.1.4 Access Requirements 2](#_Toc161674603)

[1.2 Non-Functional requirements 2](#_Toc161674604)

[1.2.1 NF-1 Performance 2](#_Toc161674605)

[1.2.2 NF-2 Scalability 3](#_Toc161674606)

[1.2.3 NF-4 Security 3](#_Toc161674607)

[1.2.4 NF-5 Usability 3](#_Toc161674608)

[1.2.5 NF-6 Maintenance 4](#_Toc161674609)

[1.3 External Interface Requirements 4](#_Toc161674610)

[1.3.1 User Interfaces (Phase 1) 4](#_Toc161674611)

[1.3.2 Hardware Interfaces 4](#_Toc161674612)

[1.3.3 Communications Interfaces 4](#_Toc161674613)

[1.3.4 Software Interfaces 5](#_Toc161674614)

[1.3.5 Application Programming Interfaces 5](#_Toc161674615)

[1.3.6 Database Interfaces 5](#_Toc161674616)

[2 UML Diagrams 6](#_Toc161674617)

[2.1 Class Diagram 6](#_Toc161674618)

[2.2 Sequence Diagram 7](#_Toc161674619)

[2.3 Use case diagrams 8](#_Toc161674620)

[2.3.1 Normal Case 8](#_Toc161674621)

[2.3.2 Error Case 9](#_Toc161674622)

[3 Test Cases 10](#_Toc161674623)

[4 User Manual (Phase 1) 11](#_Toc161674624)

[5 Compilation Instructions 12](#_Toc161674625)

[6 Peer Review Feedback 12](#_Toc161674626)

[7 Reflection 12](#_Toc161674627)

[8 Member Contribution Table 13](#_Toc161674628)

# Requirements specification for phase one

## Functional Requirements covered under phase one.

### REQ-1 Inventory Management Requirements

**REQ-1.2 Product Management**

Importance: An inventory management system's foundation is product management. Product information must be created, updated, and deleted. Businesses may guarantee precise inventory tracking, effective order processing, and well-thought-out product assortment decisions by handling product information well.   
  
Strategy for Implementation: To implement product management, a database schema supporting attributes like name, SKU, price, quantity must be in place. Product data entry and editing should be done through an intuitive interface with validation criteria to guarantee data integrity. Adding the ability to categorise and filter products will also make product management easier for users.

### REQ-4 Supplier and Customer Requirements

**REQ-4.1 Supplier Management**

Importance: Having up-to-date supplier information enables companies to handle purchase orders, negotiate better terms, and evaluate suppliers more effectively. It facilitates judgements about inventory replenishment and strategic purchasing.

Implementation Plan: Create a supplier information management module that allows users to edit, remove, and add supplier information. Functionalities like monitoring order history, and supplier details should be maintained.

User should add remove or edit a vendor information.

**REQ-4.2 Customer Management**

Significance: Customer management is essential for enhancing customer service, personalising interactions with customers, and bolstering sales and marketing plans. It supports the management of consumer preferences, the tracking of past transactions, and the identification of sales prospects.

Method of Implementation: Provided a customer profile management system with features for adding, editing, and deleting data about your customers. Targeted marketing campaigns and more individualised service will be made possible by integration with the sales order system to track order history and preferences.

Users can add, remove and edit a customer information.

### REQ-7 User Interface Requirements

**REQ 7.2 Navigation**

Significance: Users can locate the information they need quickly and easily with intuitive navigation, which is crucial for a great user experience. Effective navigation lowers the learning curve for novice users and boosts output.

Implementation Strategy: Provide intuitive, logical navigation throughout the user interface. For familiarity, we used popular user interface patterns, including sidebar navigation menus for classification. Used breadcrumbs trails to make it simple to track a user's location within an application.

### Access Requirements

**REQ-8.1 User Login/Sign Up**

Significance: Safe sign-up and login procedures are essential to user authentication, guaranteeing that only authorised users are able to access the system. This enhances user experience and safeguards critical data.

Strategy of Implementation: We have done a basic user login and signup modules which can encrypt user password and store it securely with only the authorized password and user email address, the system will allow users to access the inventory application. We will store a session key that will expire after 24 hours to make it more secure.

**REQ-8.2 User Logout**

Significance: When users terminate their sessions, a safe logout procedure guards against unwanted access to their accounts, particularly on shared or public computers.

Strategy of Implementation: Provide a feature for securely ending user sessions and clearing session data when the user logs out. Provide the user interface a clear and easy-to-access logout option. This will clear the session key inside the browser local storage.

## Non-Functional requirements

### NF-1 Performance

**NF-1.1 System Performance**

For both user pleasure and operational efficiency, it is imperative that the system be responsive and able to handle processes quickly.

Implementation Strategy: Reduce data transmission sizes, employ caching techniques, and optimise backend activities. Performance testing can be used to find bottlenecks and fix them.

### NF-2 Scalability

**NF-2.1 Managing the load.**

A business must be able to handle growing workloads without experiencing a decline in performance in order to grow.

Here is an easy method to follow: Construct the system so that it can use cloud services, which automatically scale resources to meet demand, to scale up operations.

**NF-2.2 Scalable**

When there is a high amount of data is being used or let’s say a sudden surge of users flowed in and using the app at once. Then the app should be able to handle such huge amounts of requests so the amount should be scalable.

This also helps if our business grows suddenly because of a trend that’s been going on that helps our cause of the need for being able to do so in many ways of providing a saleable app. This will capture the trendy market.

### NF-4 Security

**NF-4.1 Authentication**

In order to secure sensitive data and prevent unwanted access, robust authentication is essential.

Strategy of Implementation: Use robust authentication methods, such as Json web tokens that are used to manage authentication. Also, we have used the encoding to save the password more secure and reliable. And the JWT token will expire every 24 hours, so we have to login in again after the 24 hours mark every day.

### NF-5 Usability

**NF-5.1 Friendly UI UX**

A user-friendly interface is necessary to guarantee that users can interact with the system without difficulty or frustration.

A straightforward strategy focuses on a clean, logical layout with obvious labelling. Include users in the design process to get their feedback and make adjustments based on the outcomes of usability testing.

**NF-5.2 Accessibility**

In addition to meeting many legal requirements, making your programme accessible to people with disabilities will increase its user base.

A simple method to accomplish this is to adhere to the Web Content Accessibility Guidelines (WCAG). Incorporate functions such as keyboard navigation, contrast adjustment, and screen reader support.

### NF-6 Maintenance

**NF-6.1 Components**

The Application should be maintained as components and components such as an important task will be split into a group of individual tasks that merge together to form a bigger one.

This is useful in n number of situations, such as if say there is a bug that has to be solved within a few hours if the code is not maintained in components, it might take longer than expected to solve it.

If the code is in modular, we can solve it easily, so this is helpful to maintain the application.

## External Interface Requirements

### User Interfaces (Phase 1)

* This application has a clean and user-friendly interface that could be easily understood and accessible.
* Login/Signup page
* Items page
* Customers Page
* Vendors Page
* Side Pannel

### Hardware Interfaces

* This application needs a decent pc that could host a node server and a vite react application to run.
* And a browser parallelly to see the application in action.

### Communications Interfaces

This app will make communications over a safe http request that authenticates at the server side, every communication is validated based on the JWT token that is saved at browser.

### Software Interfaces

* HTML
* Java Script
* CSS
* React (UI Library based on java script)
* SQLite (Data base)
* JSX (java script xml used in React)
* Material UI (UI library components)
* Yarn (Package manager)
* Express (Node API request processor)

### Application Programming Interfaces

* These are the main connection between the UI and the server.
* Through the API’s we will send and receive data via an authenticated manner.

### Database Interfaces

The database interface is the one that stores all the information about the database and then we retrieve from it.

All the user data and the organization data will be stored and retrieved through various clusters of databases that are stored at the server side.

This will help the application retain the user data.

# UML Diagrams

## Class Diagram

The Phase 1 class diagram of the "Inventory Management System" lists attributes below the rectangles that represent the objects, with class names placed in the upper section. Links are used to show relationships between various objects, and lines are used to indicate the connections between these objects.

A diagram of a server

Description automatically generated

## Sequence Diagram

Sequence diagram for the "Inventory management system" in relation to the requirements for Phase 1. Objects are the entities that converse within the system. Vertical lines are used in the sequence diagram to represent them.

A diagram with text and arrows

Description automatically generated with medium confidence

## Use case diagrams.

### Normal Case

The inventory management system's typical user system interactions are depicted in this diagram, which includes tasks like product browsing, user and inventory management by administrators, and system analytics review. It demonstrates how various user roles carry out essential functions with ease, guaranteeing the effective running of the inventory management procedure.

A diagram of a product

Description automatically generated

### Error Case

The way the system handles error scenarios like unsuccessful login attempts, unauthorised access, problems with data validation, system overloads, and network or database errors—is illustrated in this diagram. With an emphasis on error detection, notification, and management techniques, it highlights the robustness of the system and guarantees a flawless user experience even in the event of disruptions.

A diagram of a user

Description automatically generated

# Test Cases

There is no explicit way to execute test cases for now we must do the test cases manually. You can follow the following things to do the actions for input and output results.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Input** | **Output** | **Test Result** |
| Check if the signup works | Name: name  Email: [email@email.com](mailto:email@email.com)  Password: 12345678  Org Name: Org | Navigates to login page | pass |
| Check if the signup works, if given a same email adress | Name: name  Email: [email@email.com](mailto:email@email.com)  Password: 12345678  Org Name: Org | Shows error | pass |
| Check if the Login works | Email: [email@email.com](mailto:email@email.com)  Password: 12345678 | Navigates to dashboard | pass |
| Check if the Login works if given wrong password | Email: [email@email.com](mailto:email@email.com)  Password: 123456 | Nothing happens | pass |
| Item add | Navigate to items and click on add new item, fill the form and create. | Item creates | pass |
| Item update | Click on the three dots on an item and select the edit icon, update the form and submit | Item updates | pass |
| Item delete | Click on the three dots on an item and select the button. | Item delete | pass |
| Customers add | Navigate to Customers and click on add new Customer, fill the form and create. | Customers creates | pass |
| Customers update | Click on the three dots on an Customers and select the edit icon, update the form and submit | Customers updates | pass |
| Customers delete | Click on the three dots on an Customer and select the button. | Customers delete | pass |
| Vendors add | Navigate to Vendors and click on add new Vendor, fill the form and create. | Vendors creates | pass |
| Vendors update | Click on the three dots on an Vendor and select the edit icon, update the form and submit | Vendors updates | pass |
| Vendors delete | Click on the three dots on an Vendor and select the button. | Vendors delete | pass |

# User Manual (Phase 1)

**System Access:**

Requirements: Internet connection, modern web browser.

URL: Access the system by entering the provided one of the URL after running the UI repository.

**Create Account:**

Click “Sign Up” on the login page, enter your details, and submit.

Make sure the email wasn’t used before.

**Logging In:**

Login: Enter your email and password on the login page and click “Login.”

**Managing Items:**

Add: Go to “Items” > “Add New Item,” fill in details, and save.

Update/Delete: Find the item under Items action by clicking on the three dots, select “Edit” to update or “Delete” to remove.

**Managing Vendors:**

Add: Navigate to “Vendors” > “Add New Vendor,” enter details, and save.

Update/Delete: Choose a vendor action by clicking on the three dots, click “Edit” to update or “Delete” to remove.

**Managing Customers:**

Add: Go to “Customers” > “Add New Customer,” fill out the form, and click “Save.”

Update/Delete: Select a customer action by clicking on the three dots, “Edit” to update their information, or “Delete” to remove them.

**Logging Out:**

Logout: Click the red button on the side panel called Logout.

# Compilation Instructions

**Deploying the server**

* Navigate to the server files.
* Install Node Package Manager (npm) first by installing node. (<https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>)
* Run the command “npm i”.
* Next Run “HOST=0.0.0.0 npm start”.

**Deploying the UI**

* Open a new terminal but don’t close the server one.
* Navigate to the UI files.
* Install yarn using “npm install --global yarn”.
* Run the command “yarn”.
* Next run “yarn run dev”.
* Open one of the links that pops up in the terminal.

The test cases should be tested manually as per the given instructions above in the test cases section.

This can be done only after running the application successfully.

# Peer Review Feedback

When we showed our Inventory Management System project to peers for their thoughts, they gave us some good advice. Here's a quick rundown of their suggestions and what we did about them:

**1.Session Management:** Peer told us it's a good idea to keep track of user logins better to make our system safer and easier to use. This means making sure someone can't just sneak into someone else's account and mess with things.

**Actions Taken:** We looked up how to handle user logins better and added a new way to keep sessions safe. Now, if you're logged in for too long, it'll automatically log you out, and it makes sure your login stays safe the whole time.

**2. Add SKU to Item Details:** Peer said it's important to include Stock Keeping Units (SKUs) for each item so we can tell them apart better, especially when we have a lot of different things to keep track of.

**Actions Taken:** We changed our computer system and the way our website looks so now you can see the SKU for each item. It makes it easier to know exactly what item you're dealing with.

**3.Count Orders for Customers and Vendors:** Peer suggested we keep track of how many orders each customer and vendor has. This helps us understand who we're doing a lot of business with.

**Actions Taken:** We made a new feature that keeps count of orders for every customer and vendor and shows it on their profiles. This way, we can see who our biggest customers and suppliers are and make better business decisions.

**4.Use Material UI for the Website:** Peer recommended using Material UI for the design parts of our website because it looks nicer and is easier to work with. It helps make everything look consistent and professional.

**Actions Taken:** We started using Material UI for our website, which made it look a lot better and made it easier for us to build nice-looking pages without having to do everything from scratch.

**5. Use Vite Instead of React CLI:** Peer suggested using Vite to set up our project is better than the old way we were doing it with React CLI because Vite is faster and more modern.

**Actions Taken:** We decided to switch to Vite for our project, and it really speed things up. We can make changes and see them happen right away, which helps us work faster and smarter.

Their advice really helped us make our Inventory Management System better, safer, and nicer to use.

# Reflection

When we reflect on our experience working on the Inventory Management System project, the Code Crafters team is quite proud of how far we have come. This adventure's first part has been both rewarding and challenging, bringing us significant victories as well as pointing out some areas where we really need to put in the work and get better.

**What went well:**

**Conditions Specification:** Mest most of the requirements for completing the Functional and nonfunctional requirements. And interfaces from the implementation.

**Implementation of Core Features:** Our group successfully created and combined inventory, supplier, and customer management with other core functionalities. These modules added, edited, and deleted information features show how our system can help businesses run more efficiently.

**User Interface Design:** We created a clear, user-friendly interface that makes the application easier to use and accessible to users with different levels of technical proficiency.

**Security Measures:** Password encryption and simple login/signup modules are the first measures taken to protect user data.

**Areas for Improvement:**

**Role-Based Access Control:** Assigning roles via the user login module is a key component that is still being developed. In order to improve security and usability, this feature is essential for guaranteeing that users can access only the functionalities pertinent to their roles.

**Automated Testing:** One drawback is the reliance on human testing. By automating this procedure, you could improve productivity, enable more thorough testing, and make it easier to find and fix errors.

# Member Contribution Table

|  |  |  |
| --- | --- | --- |
| **Member Name** | **Contribution description** | **Overall Contribution (%)** |
| Sameera | Led the development of the user interface design, focusing on improving usability and implementing accessibility features. | 12.5 |
| Varsha | Focused on backend development, particularly on enhancing security measures and implementing advanced authentication methods on the server side. | 12.5 |
| Nikhil | Worked on database design and optimization to improve system performance and scalability. | 12.5 |
| Jaswanth | Contributed to the development of the supplier and customer management modules, ensuring data integrity and usability at UI. | 12.5 |
| Varun | Enhanced the implementation of user authentication with the JWT tokens, and session management. | 12.5 |
| Shivani | Focused on implementing the API’s for the vendor and customer modules | 12.5 |
| Bindu | Oversaw the project management aspect, ensuring team collaboration, written the API’s for the Item module and user auth. | 12.5 |
| Manogna | Migrated the create react app to vite environment for the enhanced development speeds this is to speed up the app building. | 12.5 |