Project Report

On

Enterprise Resource Planning System

For

Radon Tech

By

Rohit Mahesh Kanade

Roll Number-1812020

MCA 3rd Year

RADON TECH

Sr. No 98, Plot 95, FL-03, Gajlaxmi Apts, Kothrud, Pune 38. | + 91 866 96 88 215

Certificate

This is to certify that **Mr. Rohit Mahesh Kanade,** who is pursuing his MCA from Institute of Management and Career Courses (IMCC), has successfully completed his project **"Enterprise Resource Planning System"** with us.
The project duration is from January 2021 to May 2021.

Rohit is a sincere and hardworking person and is committed to his work.

We wish him all the best in his future endeavors.

Regards,

On behalf of Radon Tech

Aniruddha Gohad,

CEO,

Radon Tech

Acknowledgement

I am very glad to take this opportunity to acknowledge all those who helped me in designing, developing and successful execution of my Project **“Enterprise Resource Planning System”.**

I would like to extend my thanks and gratitude to my project guide **Dr. Swapnaja Patwardhan** (Assistant Professor, IMCC and Class Co-ordinator) **-** Internal Guide and **Mr. Suyash Joshi -** External Guide for their valuable guidance and timely assistance throughout the development of this project.

I would also like to extend my thanks and gratitude to **Dr. Santosh Deshpande** (Director, IMCC**), Dr. Ravindra Vaidya** (HOD, IMCC) and **Dr. Manasi Bhate** (Head – Training and Placement, IMCC) for their constant help and support.

Last but not the least, I would like to thank all the teaching and non-teaching faculties for their cooperation.

- Rohit Kanade

INDEX

Chapter 1 - Introduction

1.1 Company Profile

Radon tech is a software company specializing in web and mobile applications development. Radon Tech delivers products using latest and cutting-edge technology stack.

What Radon Tech Do?

Radon Tech takes care of client’s products with keeping user experience, maintainability and performance mind.

Radon tech mainly works on:

- Web Development
- Mobile Applications

Front end:

React, Angular and Vue

Back end:

Node, Golang and .NET Core coupled with SQL and NoSQL databases

Mobile:

Flutter, Dart,

Mission:

Radon Tech's mission is to provide customer a specialized, reliable, high-quality, sophisticated services with cost saving. Our customer must experience that working with Radon Tech is more professional, less risky way to develop and implement project than working completely in-house.

Solutions:

Radon Tech envelops information solutions that enable your business users to access content from any source, seamlessly delivered to any device and with minimal disruption to your existing systems.

1.2 Existing System and Need for System

Existing System:

The process of Receiving orders and delivering products and keeping track of status is done manually

In the manual existing system following processes are done:

- Once client places their order it is entered manually in a book
- Order details are passed to the workers manually
- Once the order is sent to production to check its status, a person has to manually go to the factory and enquire the status of that specific order
- Inventory/Store is also handled manually where the data is maintained in a book which is manually entered

Need for System:

Since the current system is manual there are problems being faced by the Company:

- They have to keep details of every purchase order which again has a list of items in it which becomes very hard to maintain manually
- In manual process the status of order, moving the order to production and then to the store, everything is done manually

- Keeping records becomes a task as space is required to store all the paper on which data is entered manually
- To get any info such as order status, inventory status a person has to physically go and get the information from warehouse/factory
- Manual process is more time consuming than the automated system and reduces overall efficiency

1.3 Scope of work

Proposed system is to be implemented for the organization and deployed on their own internal server only which can be accessed within the organization only

The Scope of system can be discussed with the help of the following points:

- Displaying received order and related data in the system
- Editing purchase order data
- Searching purchase order data using specific filters
- Displaying the items that are in-production and are pending production
- Updating status of the order to keep track of the progress
- Controlling and tracking items sent to production
- Keeping track of produced items and adding items to the inventory
- Displaying all items that have finished production

1.4 Operating Environment-Hardware and Software

Hardware:

- Processor : Intel core i3 processor(Dual-core)
- RAM : 4GB
- Hard Disk : 25GB

Software:

Client Side:

- Operating System : windows 7 and above(64 bit)
- Any modern web browser(chrome,firefox etc)

Server Side:

- Node

1.5 Detail description of Technology Used:

Frontend- React.Js using TypeScript

Backend- Node.Js using Express framework and TypeScript

Database- MongoDB

React:

React is a front-end library developed by Facebook. It is used for handling the view layer for web and mobile apps. ReactJS allows us to create reusable UI components. It is currently one of the most popular JavaScript libraries and has a strong foundation and large community behind it.

ReactJS is JavaScript library used for building reusable UI components. According to React official documentation, following is the definition −

React is a library for building composable user interfaces. It encourages the creation of reusable UI components, which present data that changes over time. Lots of people use React as the V in MVC. React abstracts away the DOM from you, offering a simpler programming model and better performance. React can also render on the server using Node, and it can power native apps using React Native. React implements one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding.

React Features:

- JSX − JSX is JavaScript syntax extension. It isn't necessary to use JSX in React development, but it is recommended.
- Components − React is all about components. You need to think of everything as a component. This will help you maintain the code when working on larger scale projects.
- Unidirectional data flow and Flux − React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keeping your data unidirectional.
- License − React is licensed under the Facebook Inc. Documentation is licensed under CC BY 4.0.

React Advantages

- Uses virtual DOM which is a JavaScript object. This will improve apps performance, since JavaScript virtual DOM is faster than the regular DOM.

- Can be used on client and server side as well as with other frameworks.
- Component and data patterns improve readability, which helps to maintain larger apps.

React Limitations

- Covers only the view layer of the app, hence you still need to choose other technologies to get a complete tooling set for development.
- Uses inline templating and JSX, which might seem awkward to some developers.

NodeJs:

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v14.17.0(LTS).

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Features of Node.js

Following are some of the important features that make Node.js the first choice of software architects.

- Asynchronous and Event Driven − All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.

- Very Fast − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.

- Single Threaded but Highly Scalable − Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.

- No Buffering − Node.js applications never buffer any data. These applications simply output the data in chunks.
- License − Node.js is released under the MIT license

TypeScript:

By definition, “TypeScript is JavaScript for application-scale development.”

TypeScript is a strongly typed, object oriented, compiled language. It was designed by Anders Hejlsberg (designer of C#) at Microsoft. TypeScript is both a language and a set of tools. TypeScript is a typed superset of JavaScript compiled to JavaScript. In other words, TypeScript is JavaScript plus some additional features.

Features of TypeScript:

TypeScript is just JavaScript. TypeScript starts with JavaScript and ends with JavaScript. Typescript adopts the basic building blocks of your program from JavaScript. Hence, you only need to know JavaScript to use TypeScript. All TypeScript code is converted into its JavaScript equivalent for the purpose of execution.

TypeScript supports other JS libraries. Compiled TypeScript can be consumed from any JavaScript code. TypeScript-generated JavaScript can reuse all of the existing JavaScript frameworks, tools, and libraries.

JavaScript is TypeScript. This means that any valid .js file can be renamed to .ts and compiled with other TypeScript files.

TypeScript is portable. TypeScript is portable across browsers, devices, and operating systems. It can run on any environment that JavaScript runs on. Unlike its counterparts, TypeScript doesn’t need a dedicated VM or a specific runtime environment to execute.

Why Use TypeScript?

TypeScript is superior to its other counterparts like CoffeeScript and Dart programming languages in a way that TypeScript is extended JavaScript. In contrast, languages like Dart, CoffeeScript are new languages in themselves and require language-specific execution environment.

The benefits of TypeScript include −

Compilation − JavaScript is an interpreted language. Hence, it needs to be run to test that it is valid. It means you write all the codes just to find no output, in case there is an error. Hence, you have to spend hours trying to find bugs in the code. The TypeScript transpiler provides the error-checking feature. TypeScript will compile the code and generate compilation errors, if it finds some sort of syntax errors. This helps to highlight errors before the script is run.

Strong Static Typing − JavaScript is not strongly typed. TypeScript comes with an optional static typing and type inference system through the TLS (TypeScript Language Service). The type of a variable, declared with no type, may be inferred by the TLS based on its value.

TypeScript **supports type definitions** for existing JavaScript libraries. TypeScript Definition file (with .d.ts extension) provides definition for external JavaScript libraries. Hence, TypeScript code can contain these libraries.

TypeScript **supports Object Oriented Programming** concepts like classes, interfaces, inheritance, etc.

MongoDB:

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++.It is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Advantages of MongoDB over RDBMS:

- Schema less − MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
- Structure of a single object is clear.
- No complex joins.
- Deep query-ability. MongoDB supports dynamic queries on

documents using a document-based query language that's nearly as powerful as SQL.

- Tuning.
- Ease of scale-out − MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) working set, enabling faster access of data.

Why Use MongoDB?

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

- Professional support by MongoDB

Where to Use MongoDB?

- Big Data
- Content Management and Delivery
- Mobile and Social Infrastructure
- User Data Management
- Data Hub

Chapter 2 - Proposed System

2.1 Proposed system:

ERP(Enterprise Resource Management) system is a web based application designed to ease the process of receiving orders from customers/clients and enabling easier communication between Company's front desk handlers and Factory management staff

This ERP system is designed to manage, maintain and access the information of the Orders easily without accessing any book records/files.
The main purpose is to make Resource management of the organization hassle free , efficient and easier to maintain End users of this application will be only employees of that organization which are:

- Office area workers
- Factory workers

2.2 Objectives of System:

Because of the process being entirely manual there are issues which the organization faces due to lack of easy management

Having a centralized system will eliminate most of these problems by providing essential features like:

- No manual book keeping work for any process will be there.
- Managing inventory and keeping track of it will become easier, which will help to gather information and approve orders faster
- It will be easier for organization to keep track of received orders, status of orders in production and dispatched orders
- User interfaces are designed in such a way that end users should not need to learn any new thing to handle the system.
- Maintaining records and history should be strong enough and flexible to handle large amount of data.
- Less possibility of faulty data due to strong validations implemented both on front end and backend

2.3 User Requirements:

Navigation:

Website navigation will be done using a sidebar which will include the links for different pages on the website

Purchase order:

- A page of existing purchase orders having filters and pagination
- User is able to delete a purchase order
- Clicking on the edit button will route to the edit purchase order page
- Fields are populated with the data of the selected purchase order and a table is displayed which includes the items and their details in the purchase order
- User is able to edit and delete every item of the purchase order
- User is also able to add new items to the purchase order

Production:

- This page will have 2 sections which are, pending production and In-production which will display items according to their status

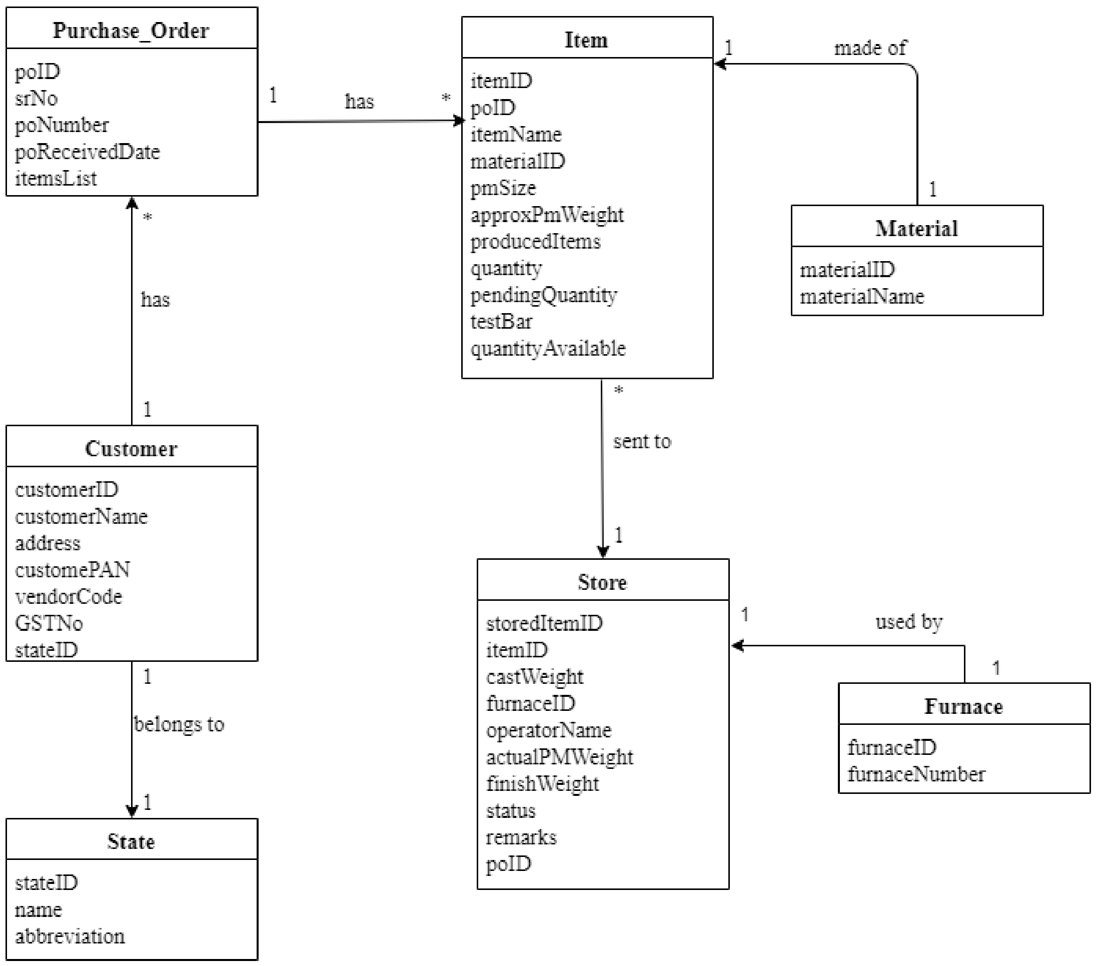
- Once the item is ready, the finished details are filled using a form and the item is sent to the store

Store:

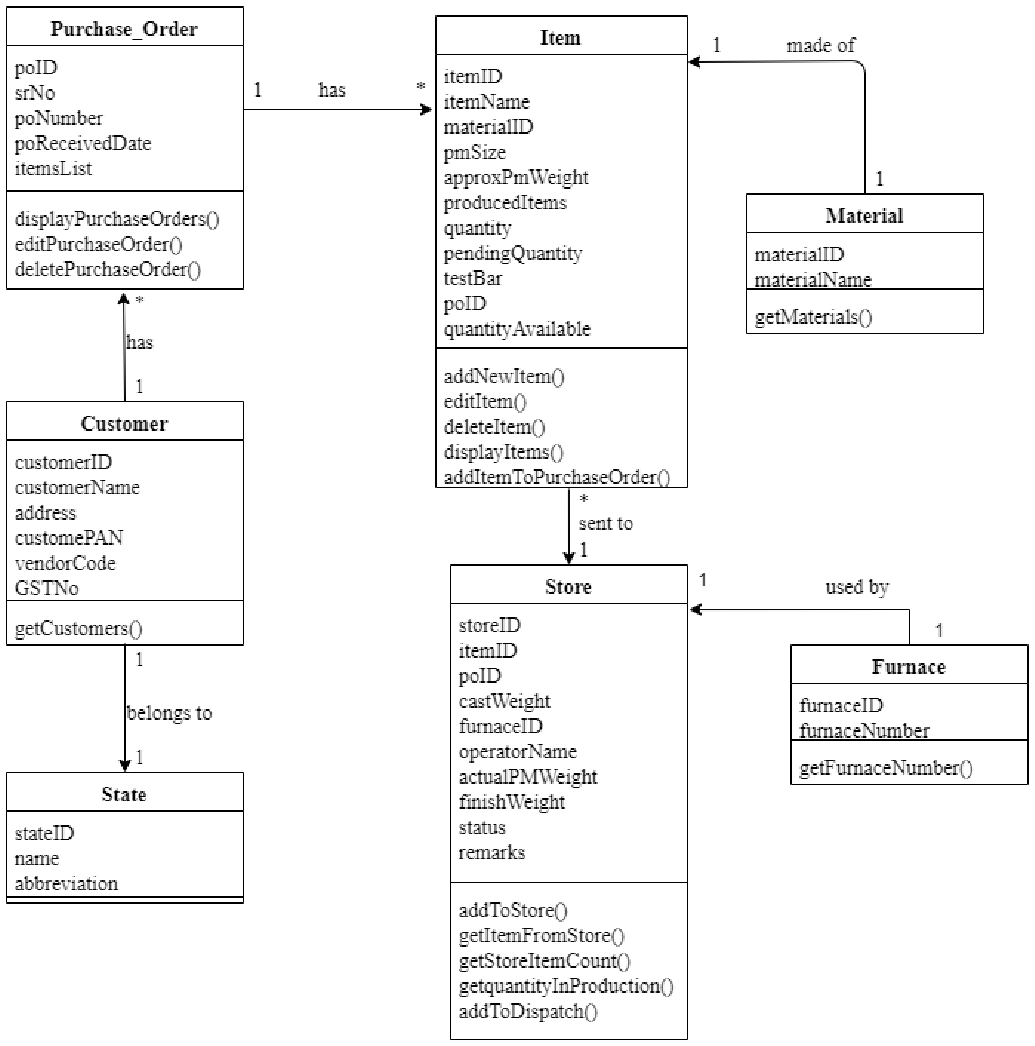
- Store will display items which have finished production
- User is able to select items and create a dispatch list

Chapter 3 – Analysis and Design

3.1 Object Diagram

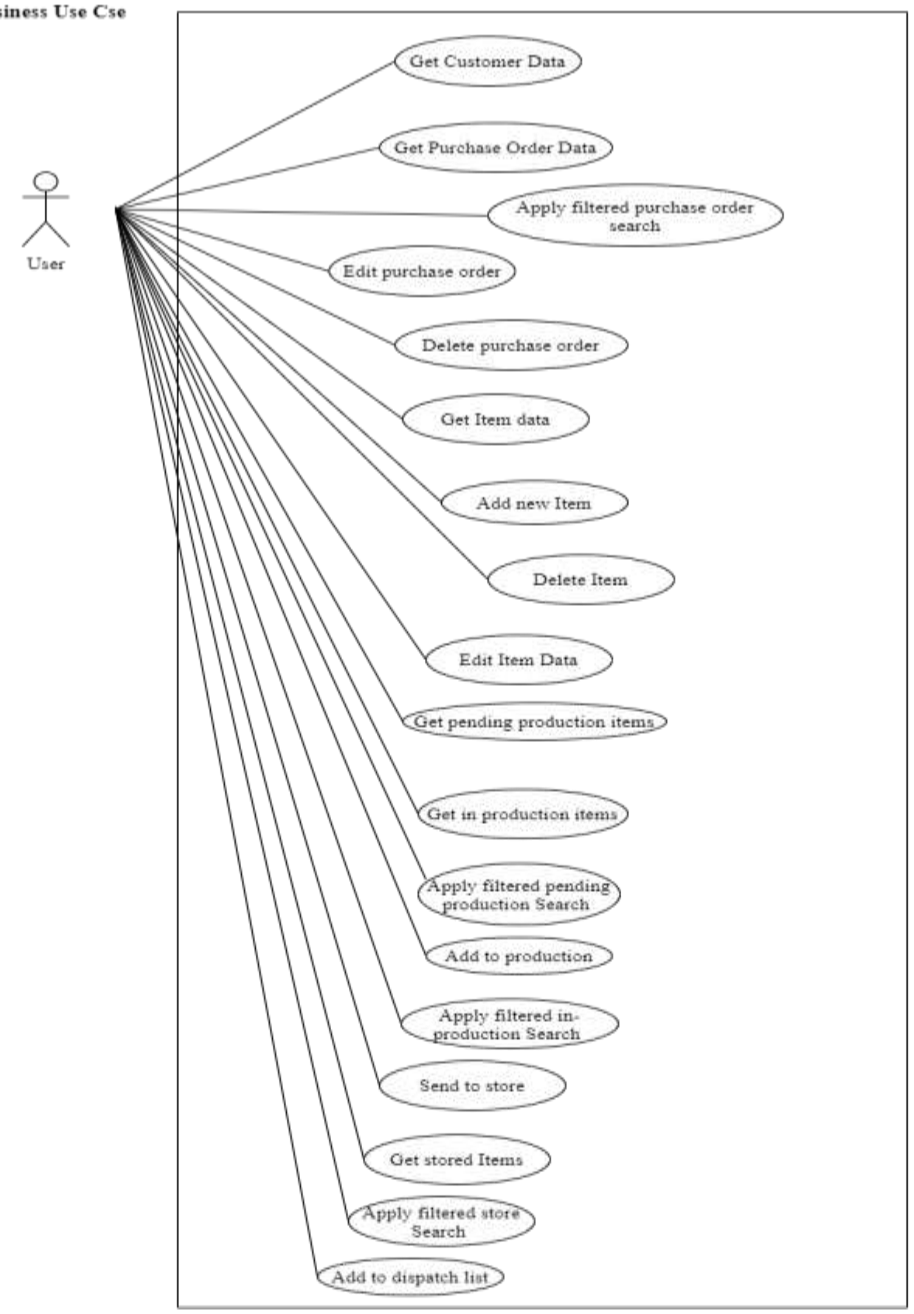


3.2 Class Diagram

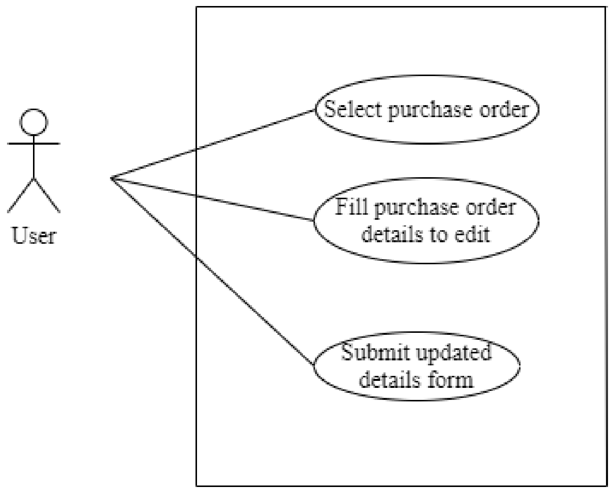


3.3 Use Case Diagram

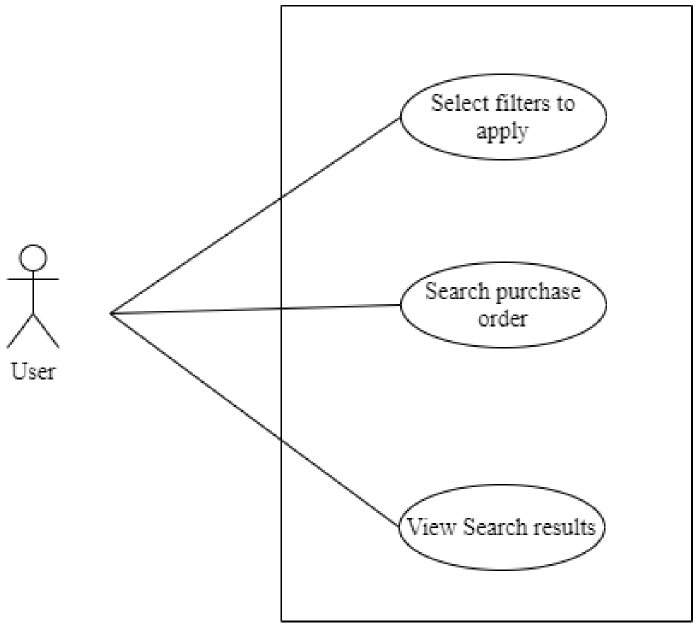
1)Business Use Cse



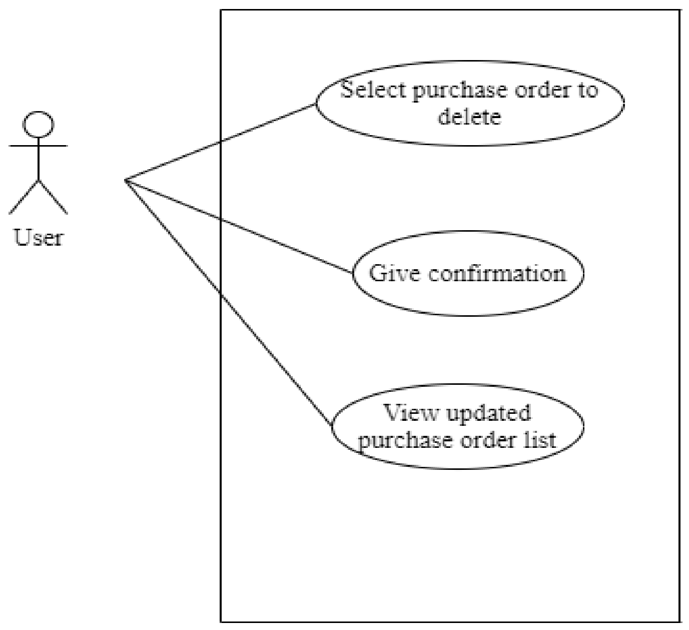
2)Edit purchase order use case


User
Select purchase order
Fill purchase order details to edit
Submit updated details form

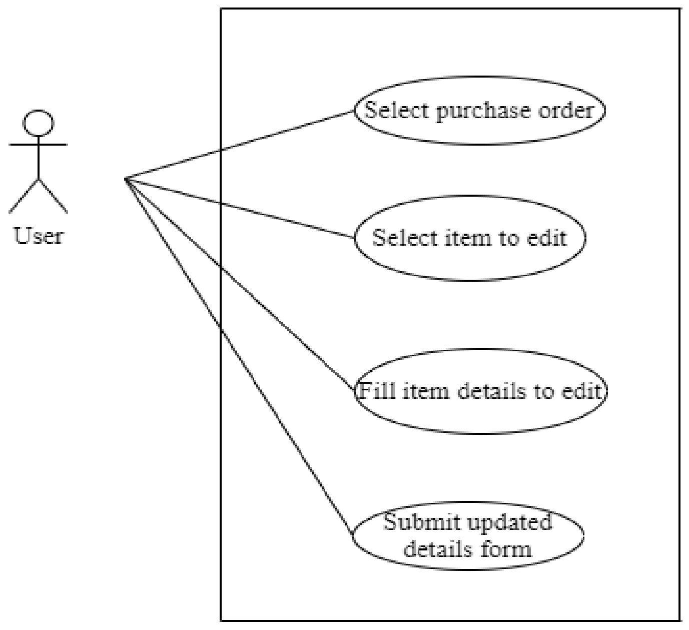
3)Search Purchase Order Use Case



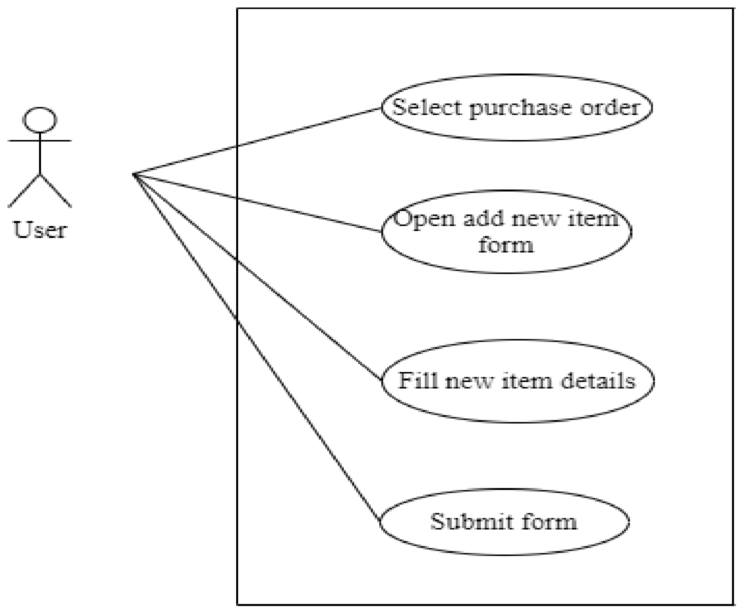
4)Delete Purchase Order Use Case


User
Select purchase order to delete
Give confirmation
View updated purchase order list

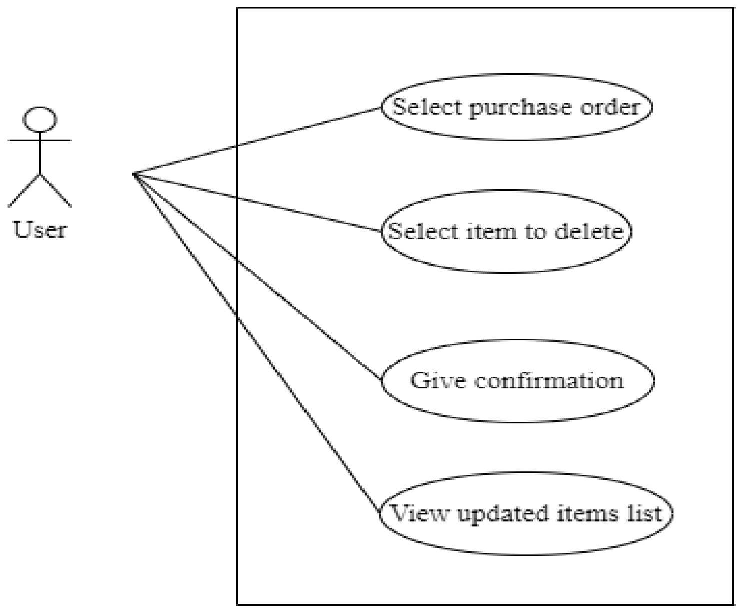
5)Edit item use case



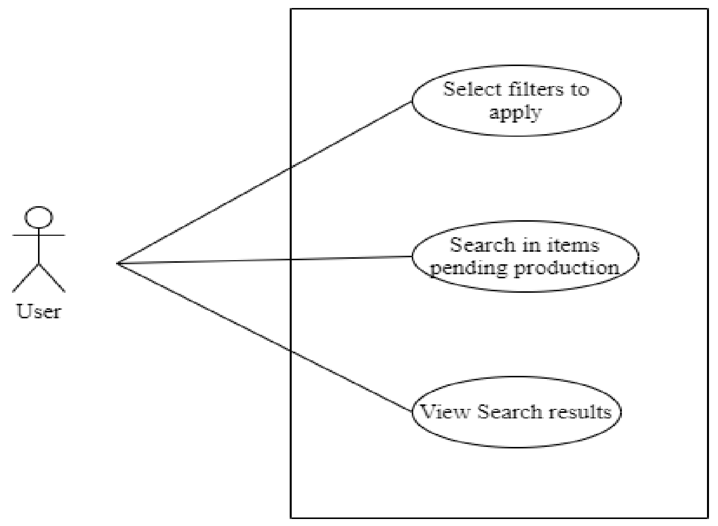
6)Add new item use case



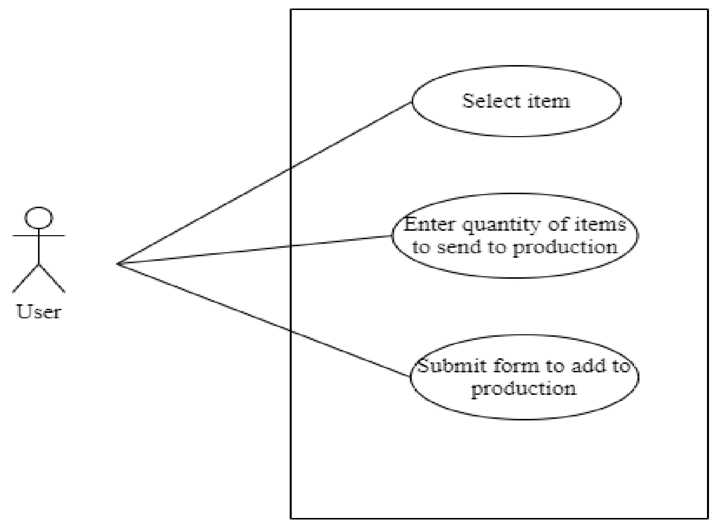
7)Delete item use case



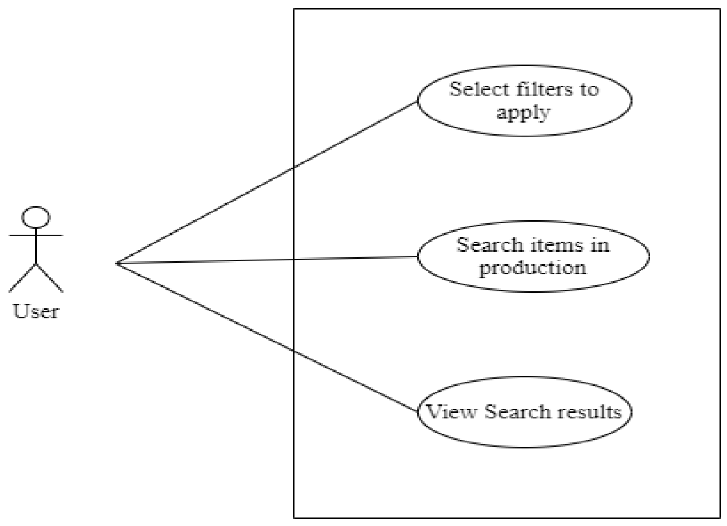
8)Search items pending production Use Case



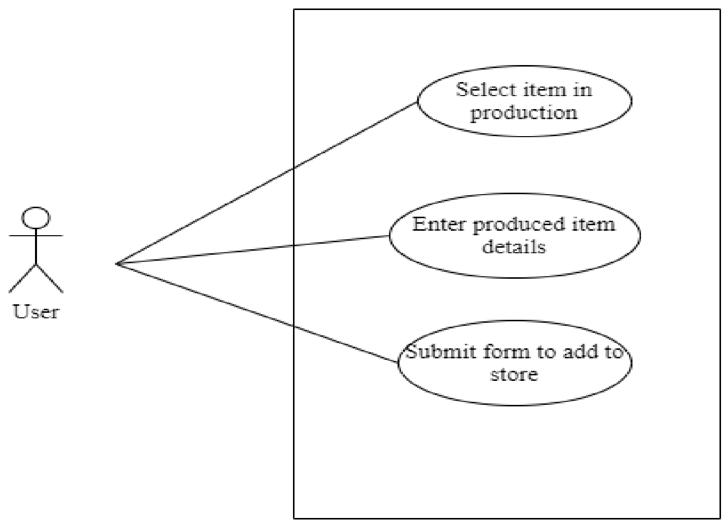
9)Add item to production use case


Select item
Enter quantity of items to send to production
Submit form to add to production
User

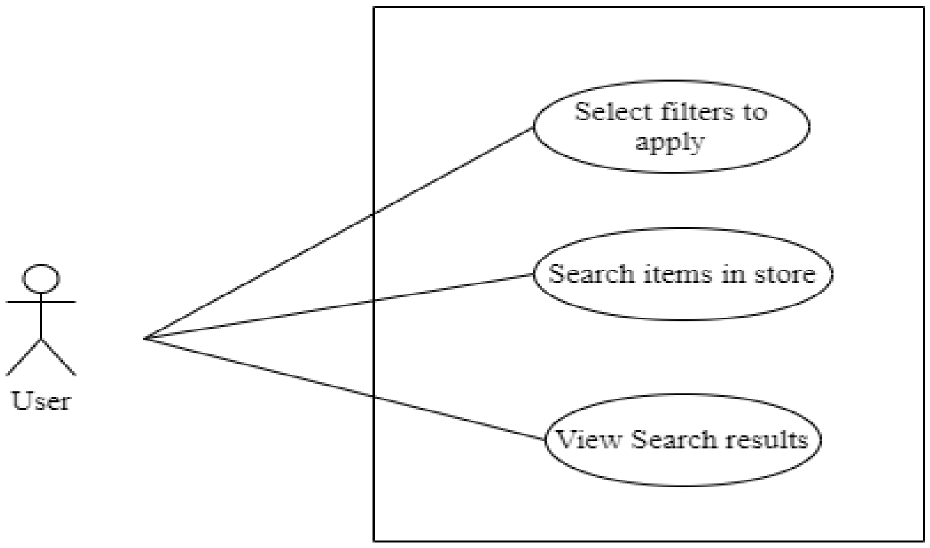
10)Search items in production Use Case



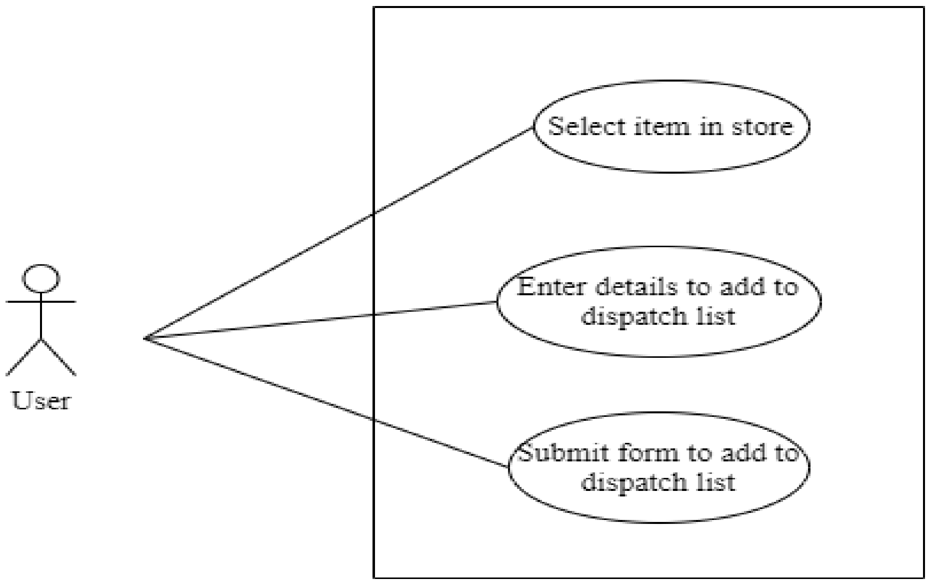
11)Send item to store use case


Select item in production
Enter produced item details
Submit form to add to store
User

12)Search items in store Use Case

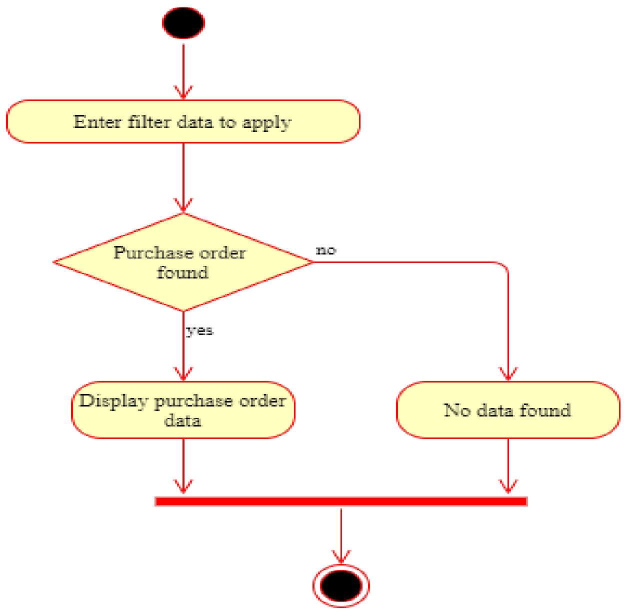


13)Add item to distpacth list use case

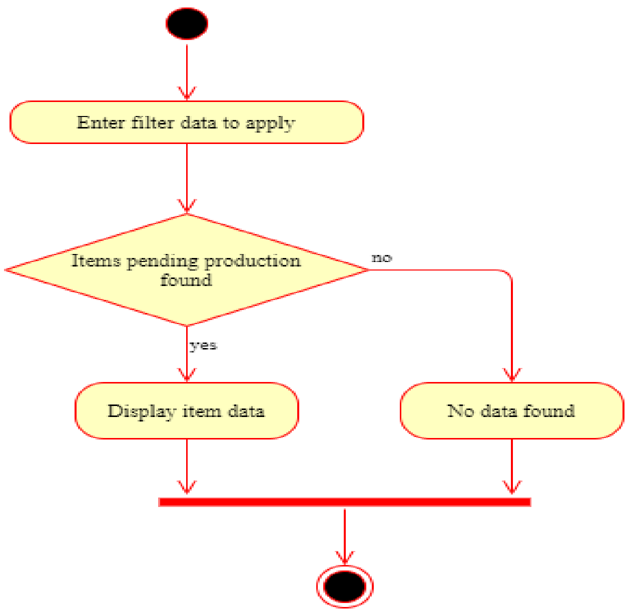

Select item in store
Enter details to add to dispatch list
Submit form to add to dispatch list
User

3.4 Activity Diagrams

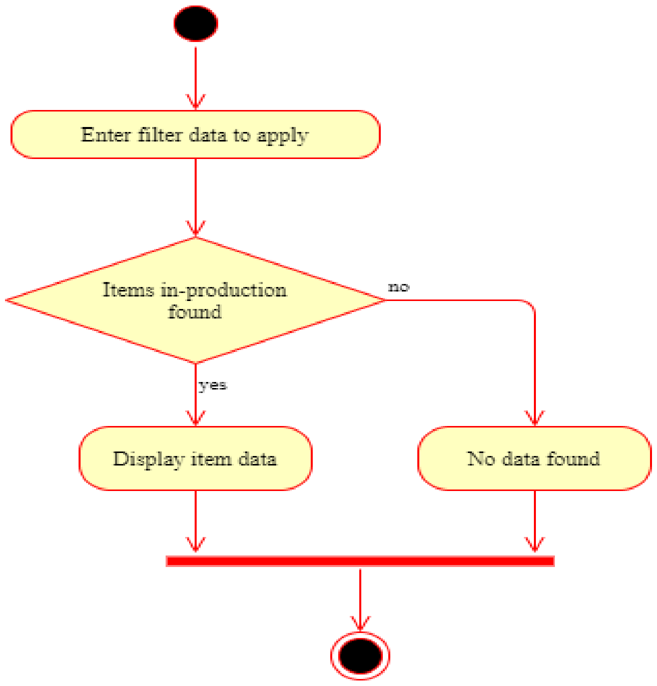
1)Search purchase orders



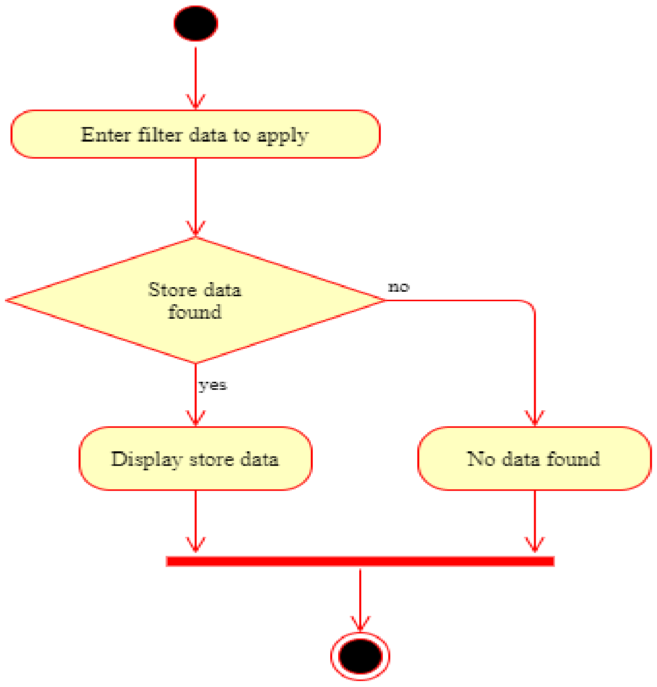
2)Search items pending production



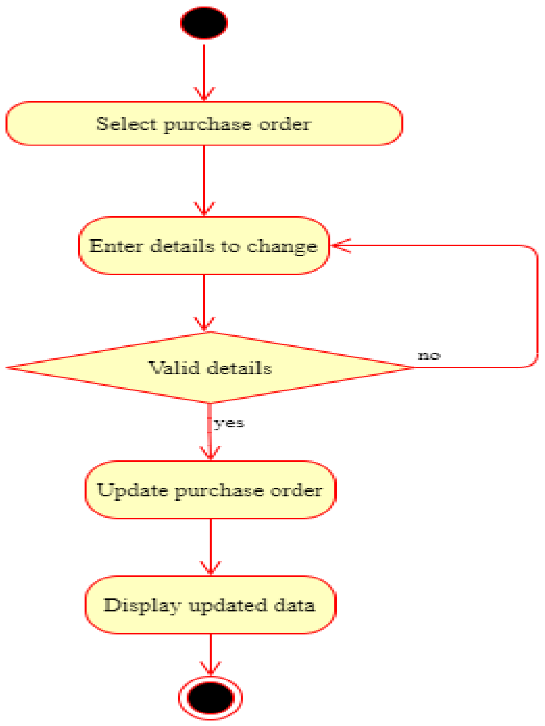
3)Search items in-production



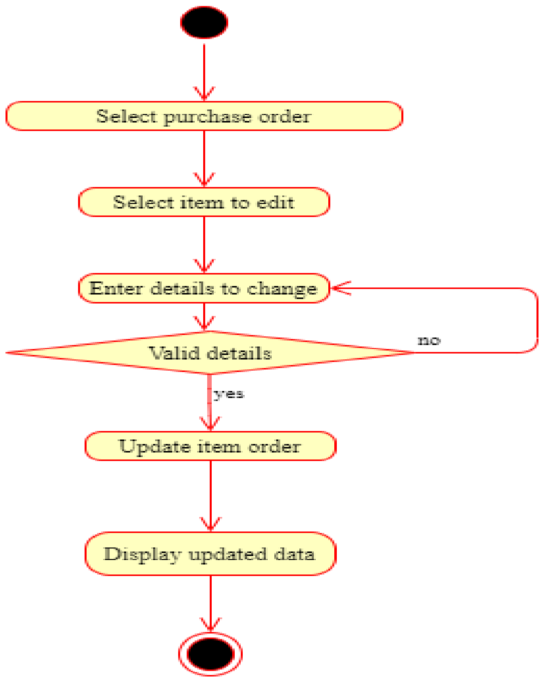
4)Search store activity



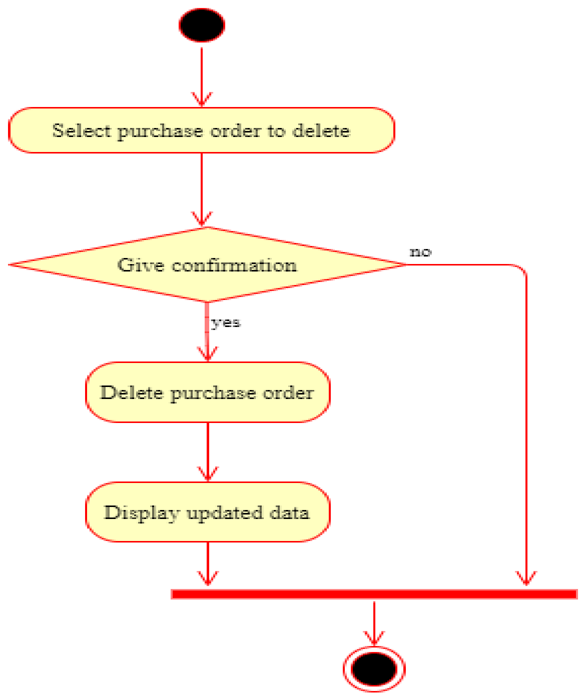
5)Edit purchase order activity diagram



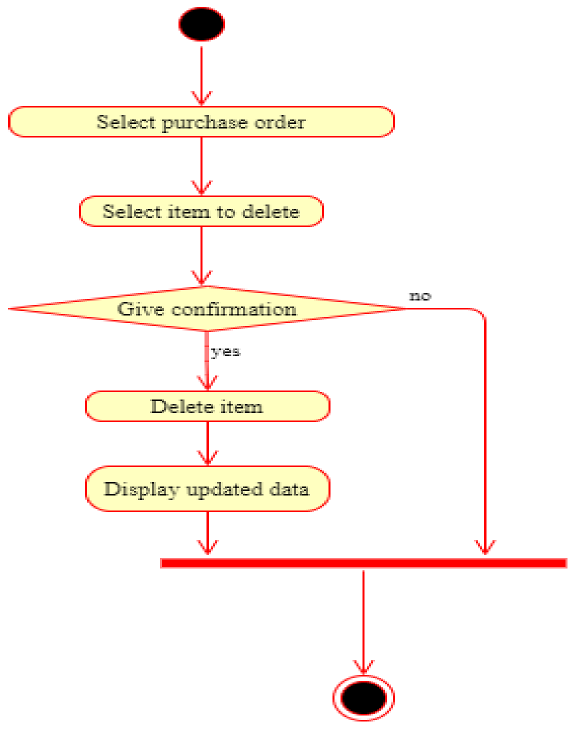
6)Edit item activity diagram



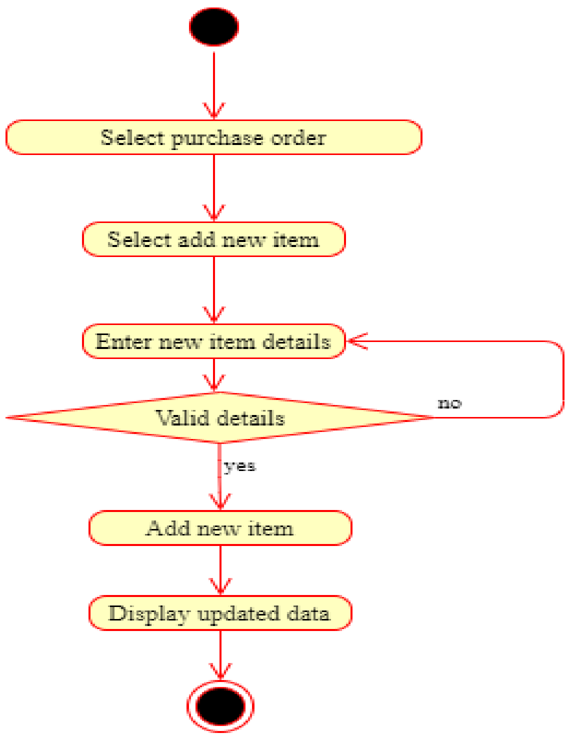
7)Delete purchase order activity diagram



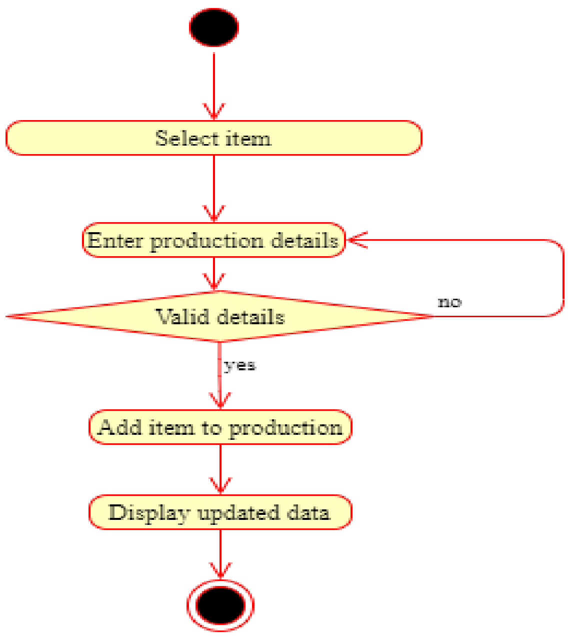
8)Delete item activity diagram



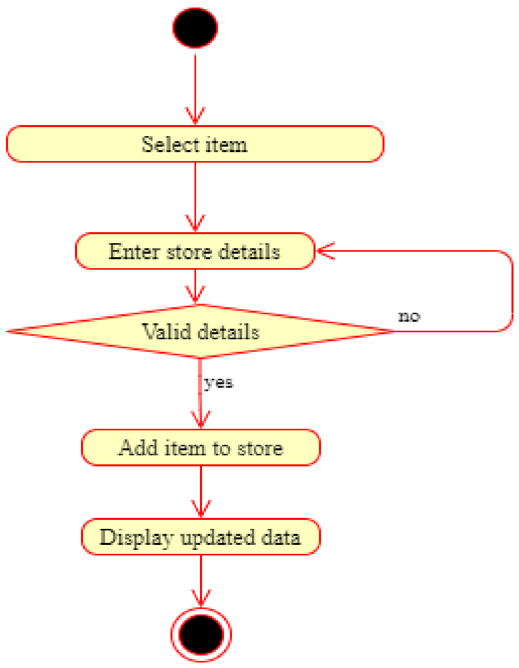
9)Add new item activity diagram



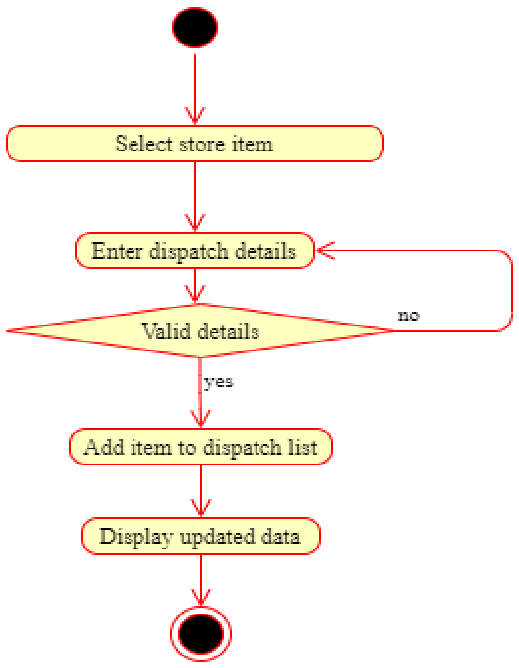
10)Add item to production activity diagram



11)Add item to store activity diagram

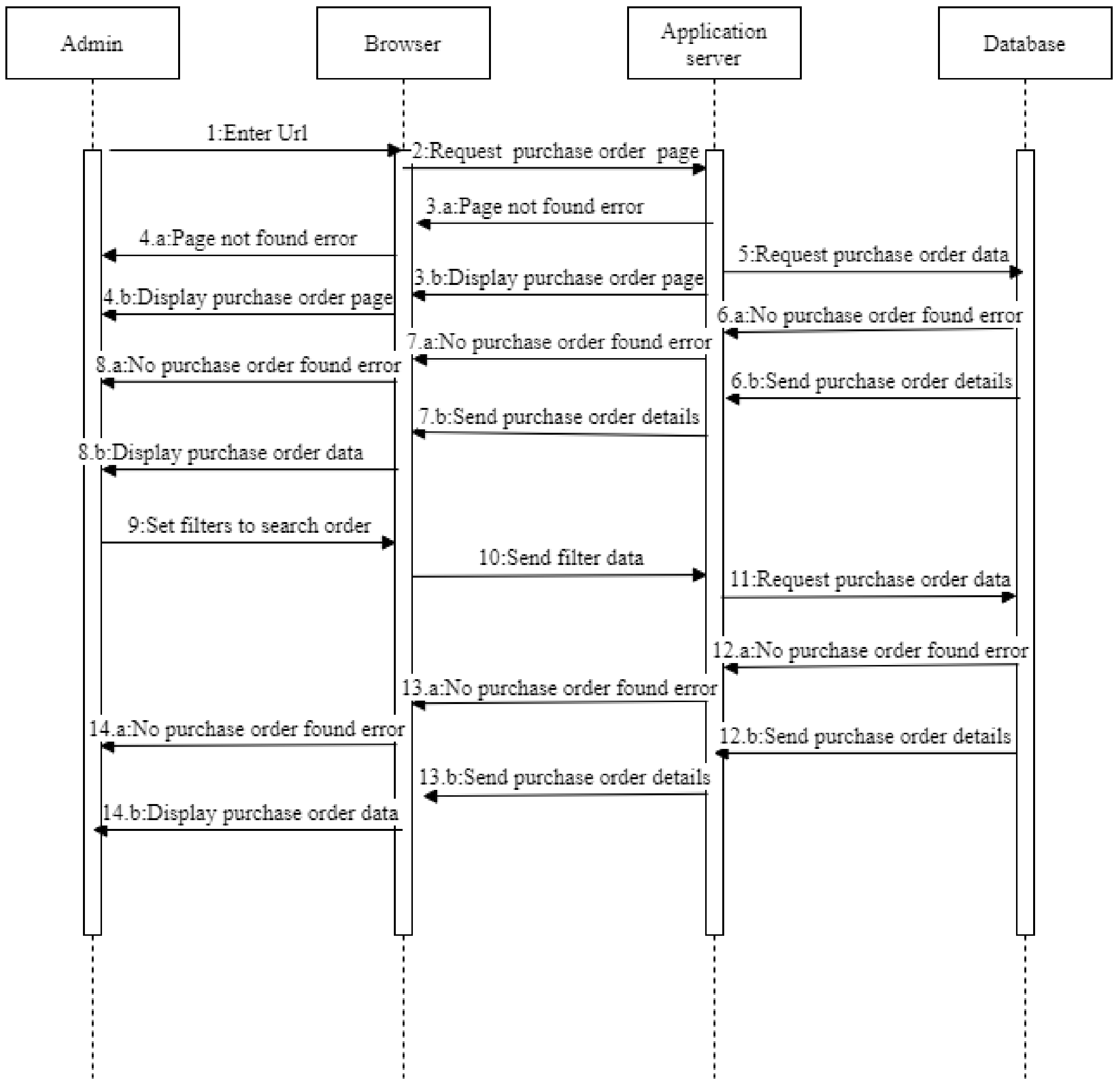


12)Add to dispatch list activity diagram

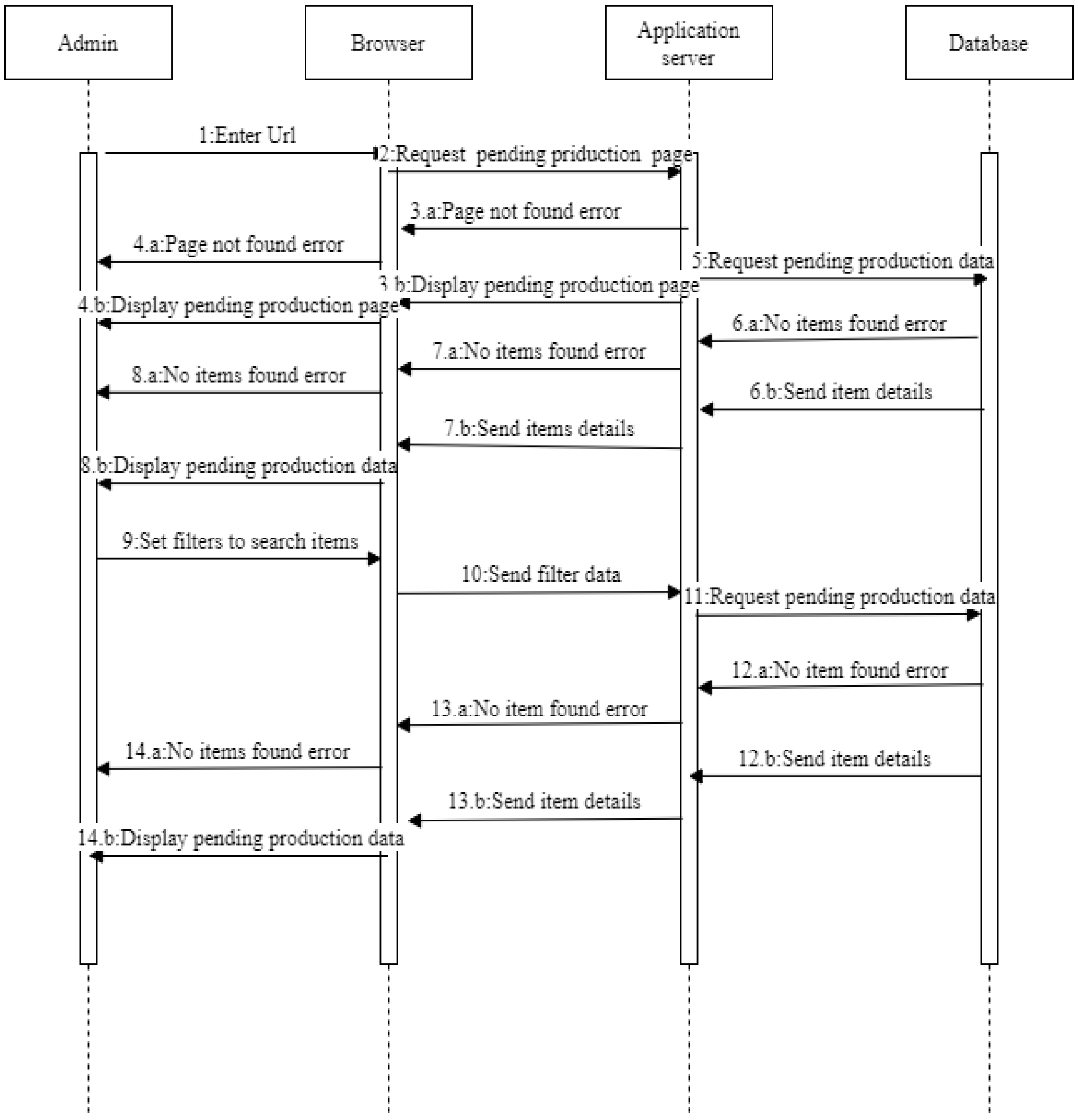


3.5 Sequence Diagram

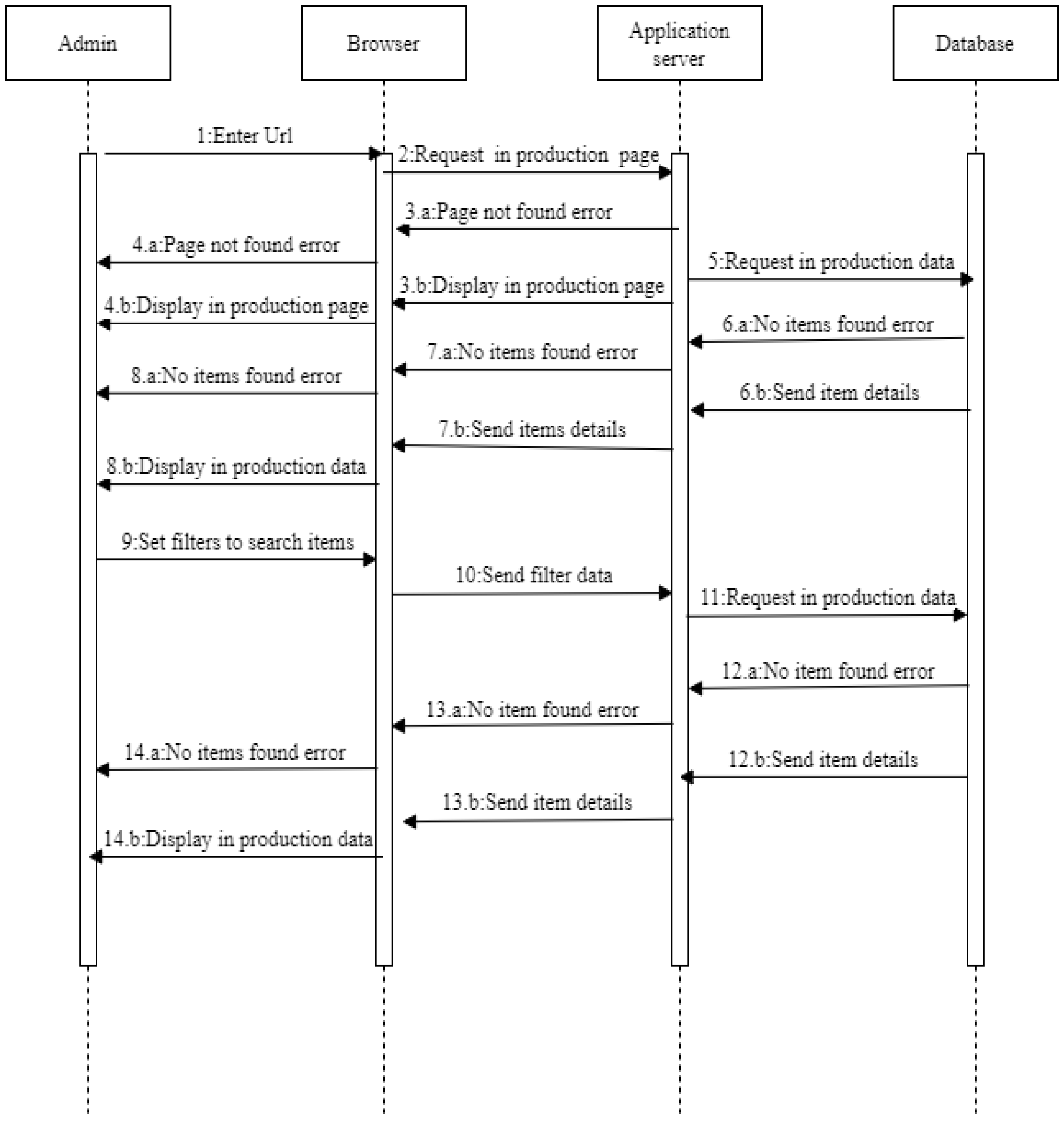
1)Search Purchase order sequence diagram


Admin
Browser
Application server
Database
1:Enter Url
2:Request purchase order page
3.a:Page not found error
4.a:Page not found error
5:Request purchase order data
3.b:Display purchase order page
4.b:Display purchase order page
6.a:No purchase order found error
7.a:No purchase order found error
8.a:No purchase order found error
6.b:Send purchase order details
7.b:Send purchase order details
8.b:Display purchase order data
9:Set filters to search order
10:Send filter data
11:Request purchase order data
12.a:No purchase order found error
13.a:No purchase order found error
14.a:No purchase order found error
12.b:Send purchase order details
13.b:Send purchase order details
14.b:Display purchase order data

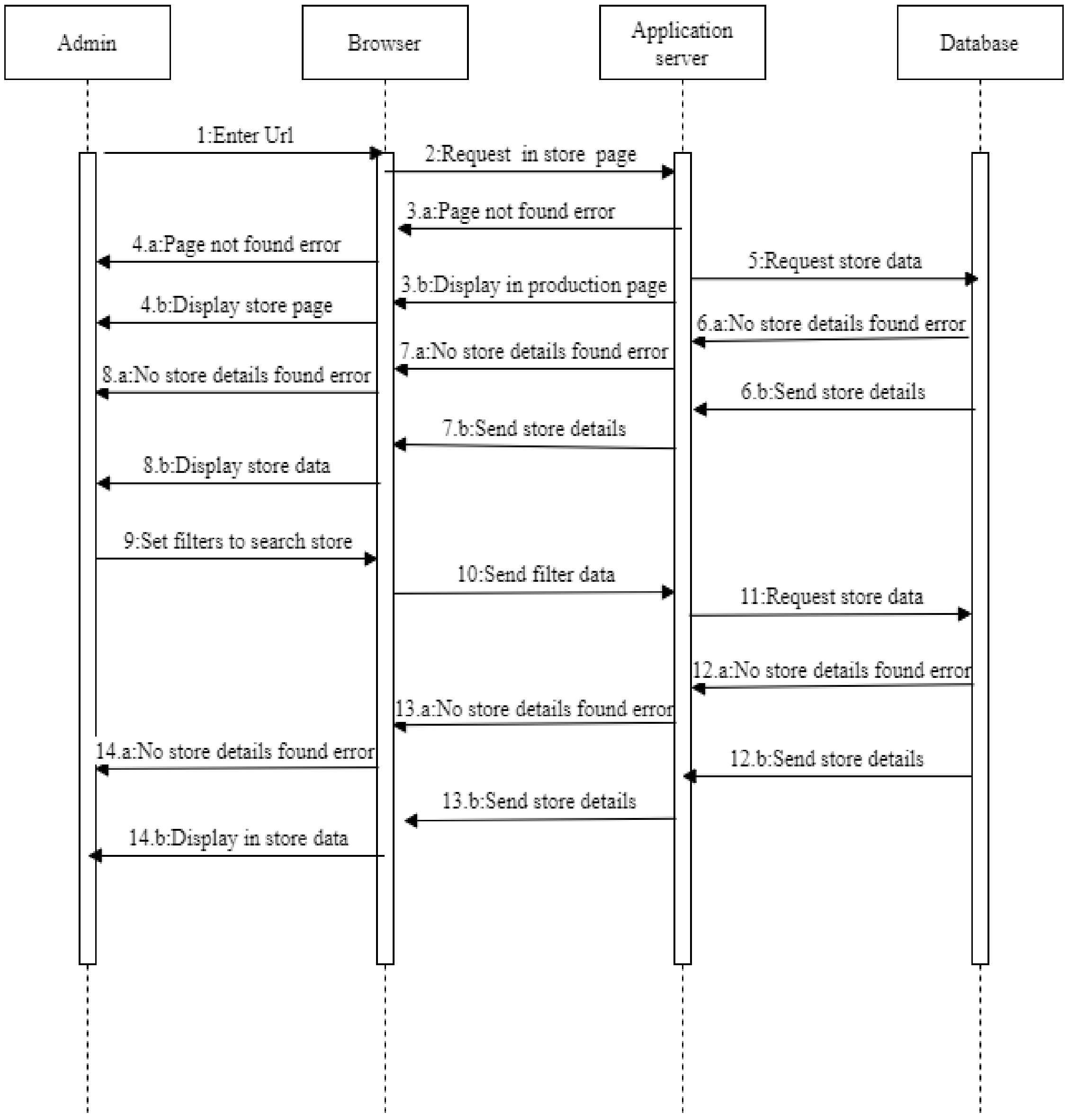
2)Search items pending production sequence diagram



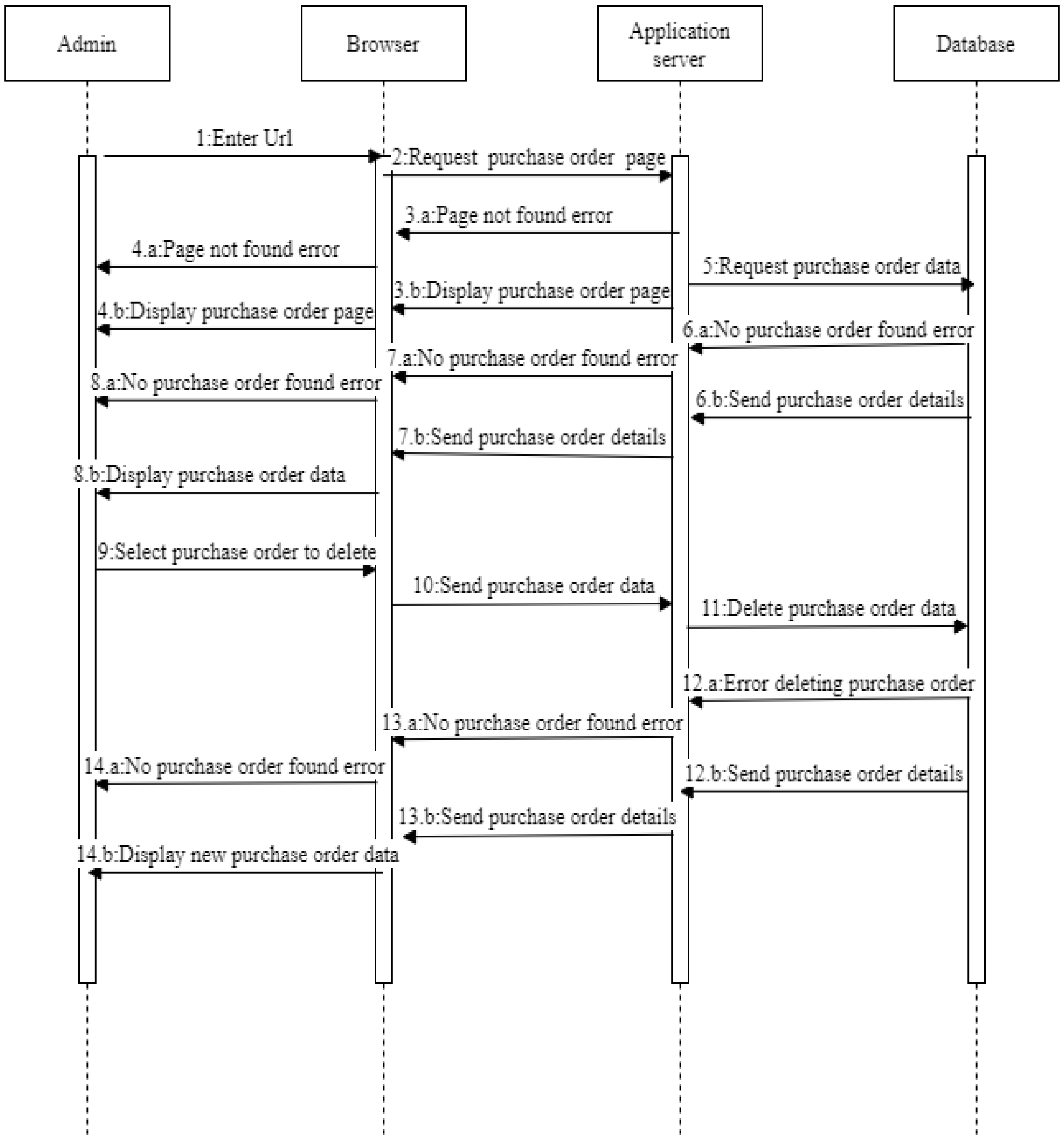
3)Search items in production sequence diagram



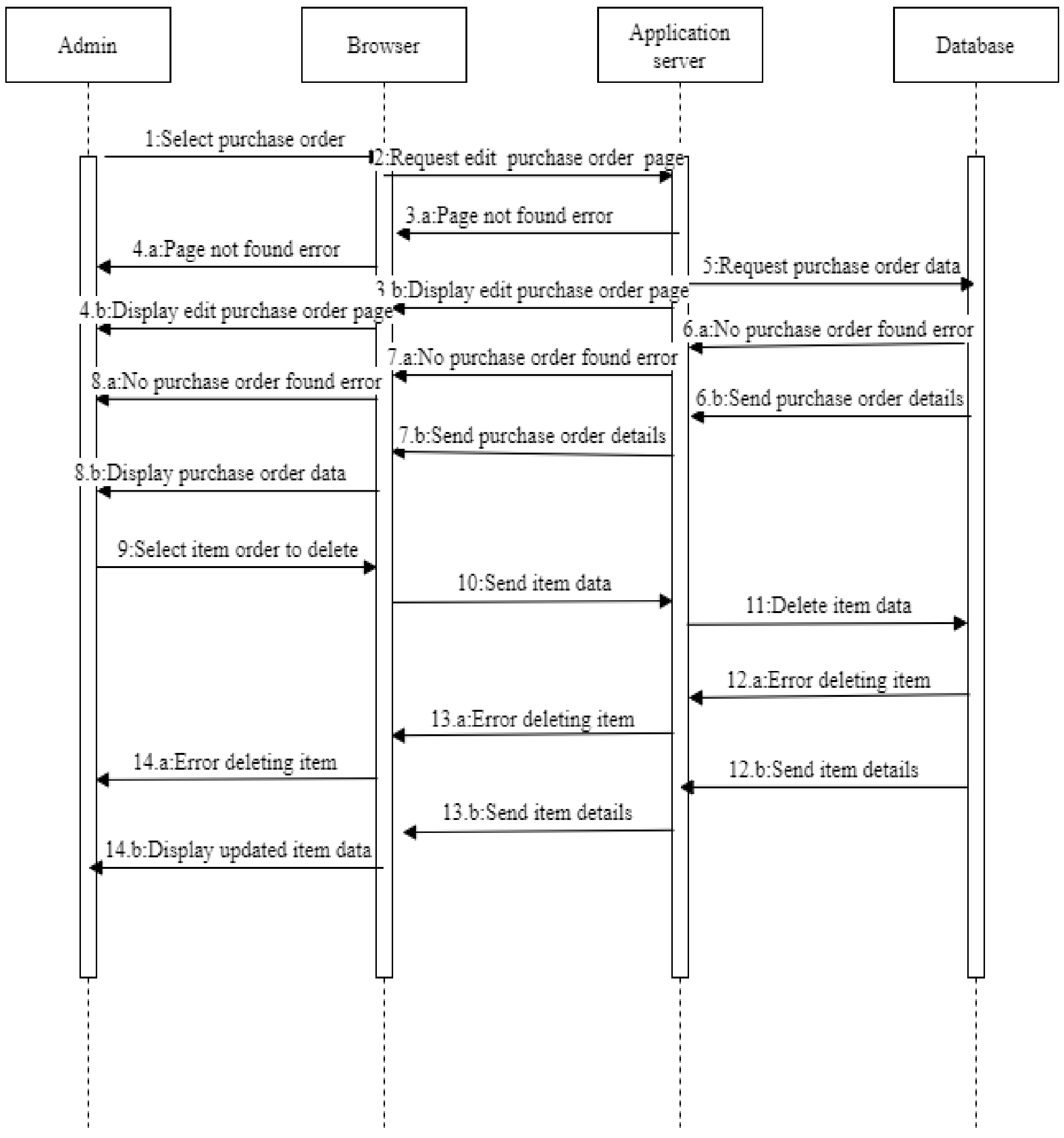
4)Search store sequence diagram



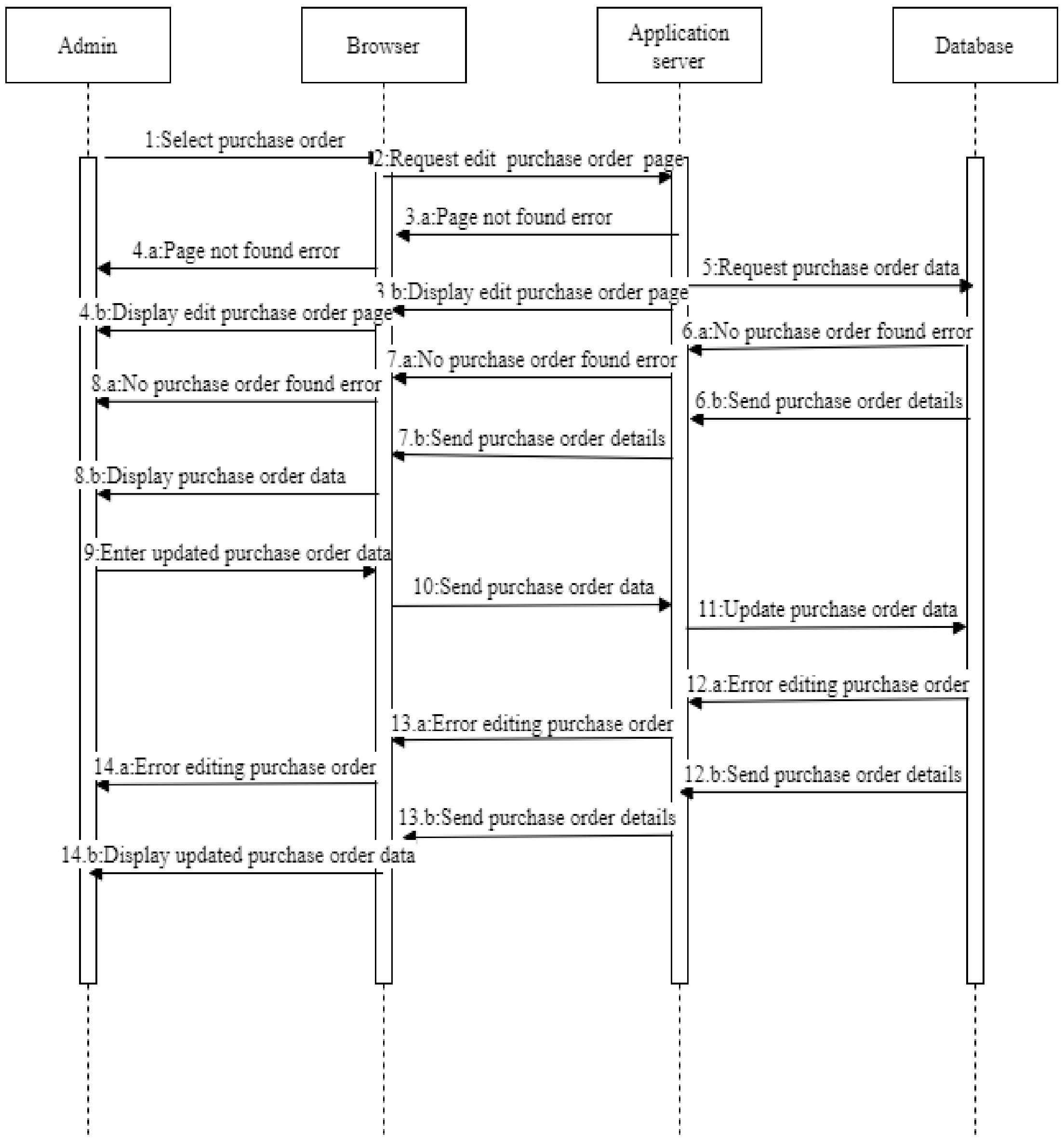
5)Delete purchase order sequence diagram


Admin
Browser
Application server
Database
1:Enter Url
2:Request purchase order page
3.a:Page not found error
4.a:Page not found error
5:Request purchase order data
3.b:Display purchase order page
4.b:Display purchase order page
6.a:No purchase order found error
7.a:No purchase order found error
8.a:No purchase order found error
6.b:Send purchase order details
7.b:Send purchase order details
8.b:Display purchase order data
9:Select purchase order to delete
10:Send purchase order data
11:Delete purchase order data
12.a:Error deleting purchase order
13.a:No purchase order found error
14.a:No purchase order found error
12.b:Send purchase order details
13.b:Send purchase order details
14.b:Display new purchase order data

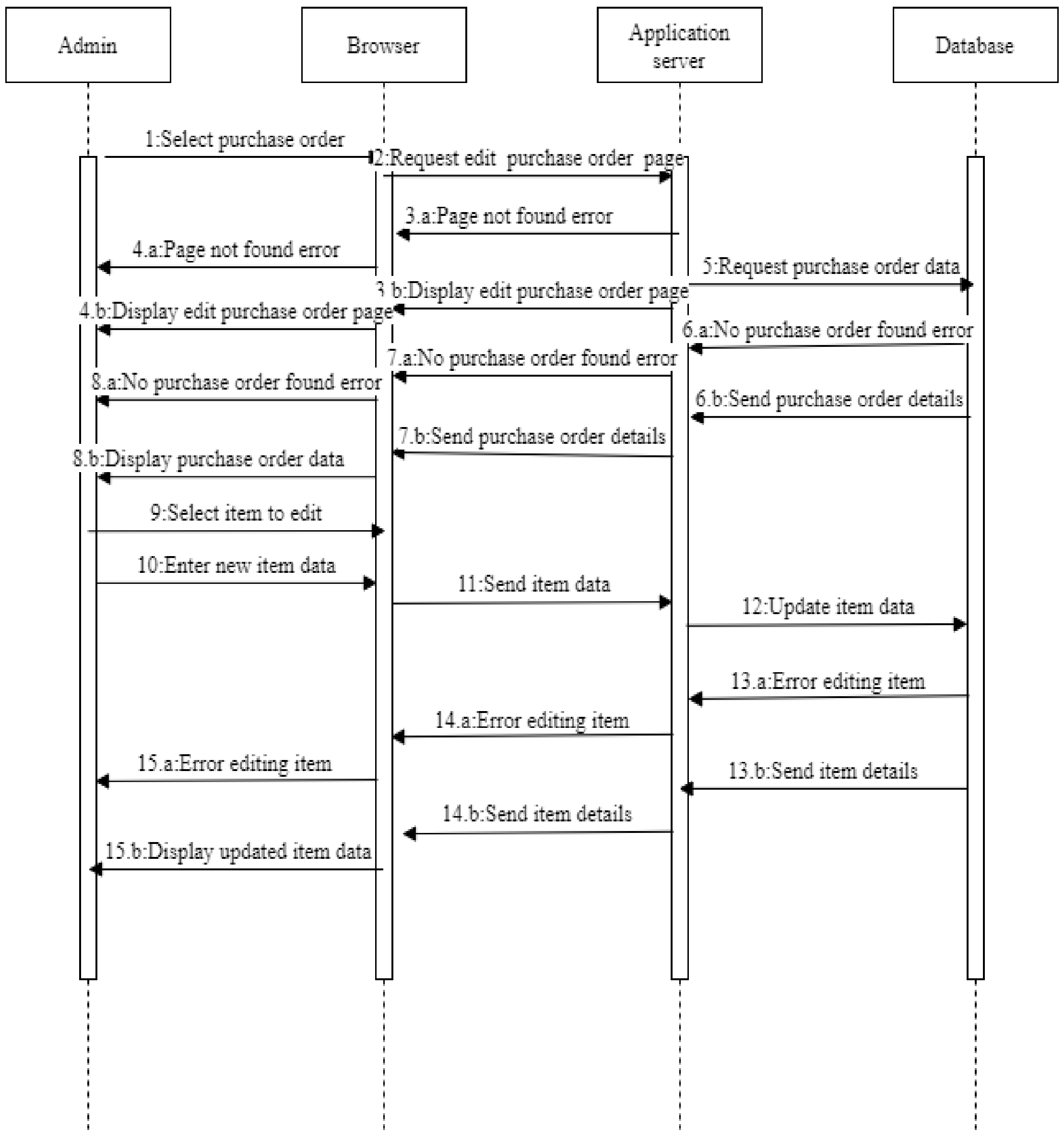
6)Delete Item sequence diagram


Admin
Browser
Application server
Database
1:Select purchase order
2:Request edit purchase order page
3.a:Page not found error
4.a:Page not found error
5:Request purchase order data
3.b:Display edit purchase order page
4.b:Display edit purchase order page
6.a:No purchase order found error
7.a:No purchase order found error
8.a:No purchase order found error
6.b:Send purchase order details
7.b:Send purchase order details
8.b:Display purchase order data
9:Select item order to delete
10:Send item data
11:Delete item data
12.a:Error deleting item
13.a:Error deleting item
14.a:Error deleting item
12.b:Send item details
13.b:Send item details
14.b:Display updated item data

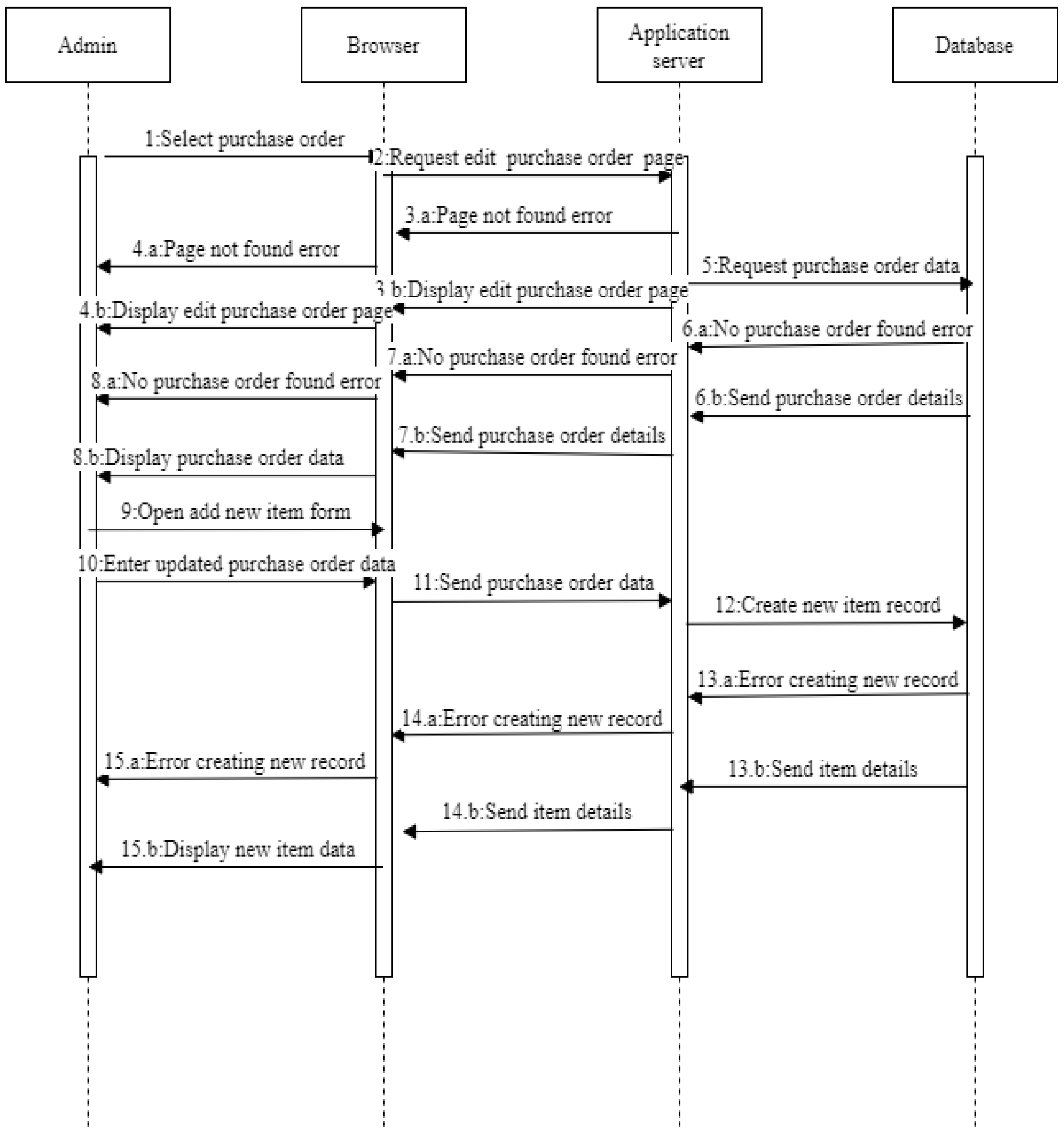
7)Edit purchase order sequence diagram


Admin
Browser
Application server
Database
1:Select purchase order
2:Request edit purchase order page
3.a:Page not found error
4.a:Page not found error
5:Request purchase order data
3.b:Display edit purchase order page
4.b:Display edit purchase order page
6.a:No purchase order found error
7.a:No purchase order found error
8.a:No purchase order found error
6.b:Send purchase order details
7.b:Send purchase order details
8.b:Display purchase order data
9:Enter updated purchase order data
10:Send purchase order data
11:Update purchase order data
12.a:Error editing purchase order
13.a:Error editing purchase order
14.a:Error editing purchase order
12.b:Send purchase order details
13.b:Send purchase order details
14.b:Display updated purchase order data

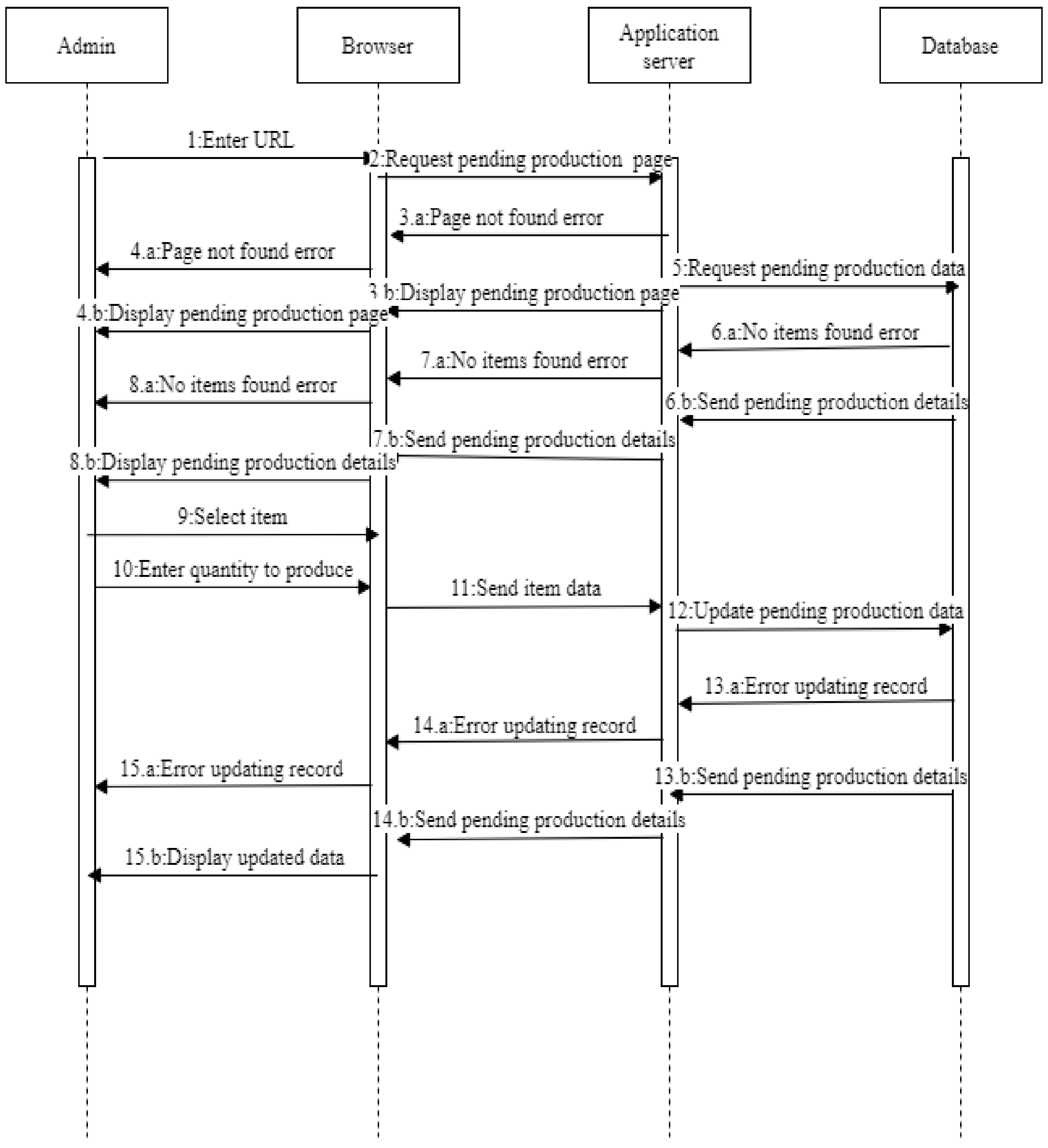
8)Edit item sequence diagram


Admin
Browser
Application server
Database
1:Select purchase order
2:Request edit purchase order page
3.a:Page not found error
4.a:Page not found error
5:Request purchase order data
3.b:Display edit purchase order page
4.b:Display edit purchase order page
6.a:No purchase order found error
7.a:No purchase order found error
8.a:No purchase order found error
6.b:Send purchase order details
7.b:Send purchase order details
8.b:Display purchase order data
9:Select item to edit
10:Enter new item data
11:Send item data
12:Update item data
13.a:Error editing item
14.a:Error editing item
15.a:Error editing item
13.b:Send item details
14.b:Send item details
15.b:Display updated item data

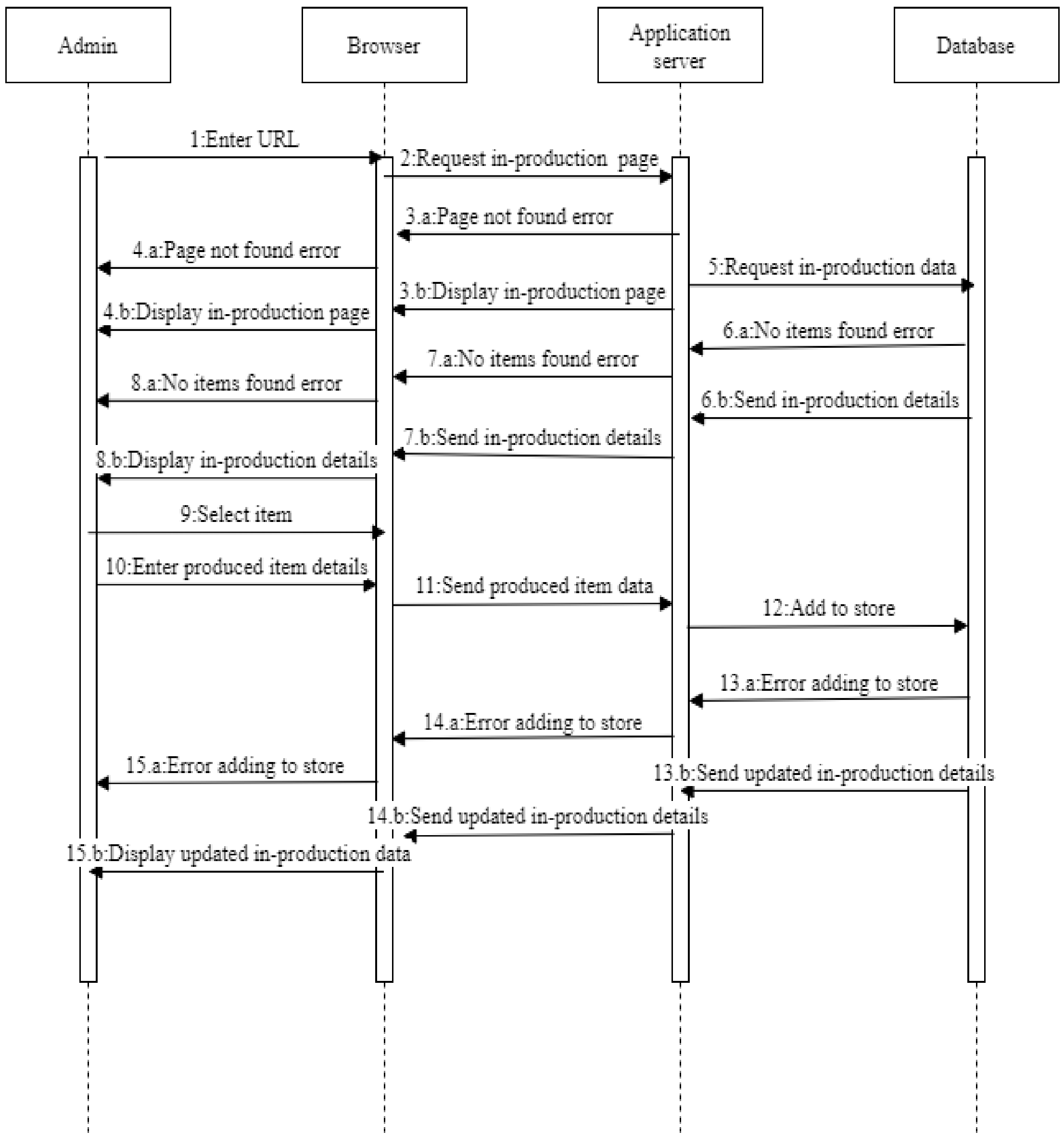
9)Add new item sequence diagram


Admin
Browser
Application server
Database
1:Select purchase order
2:Request edit purchase order page
3.a:Page not found error
4.a:Page not found error
5:Request purchase order data
3.b:Display edit purchase order page
4.b:Display edit purchase order page
6.a:No purchase order found error
7.a:No purchase order found error
8.a:No purchase order found error
6.b:Send purchase order details
7.b:Send purchase order details
8.b:Display purchase order data
9:Open add new item form
10:Enter updated purchase order data
11:Send purchase order data
12:Create new item record
13.a:Error creating new record
14.a:Error creating new record
15.a:Error creating new record
13.b:Send item details
14.b:Send item details
15.b:Display new item data

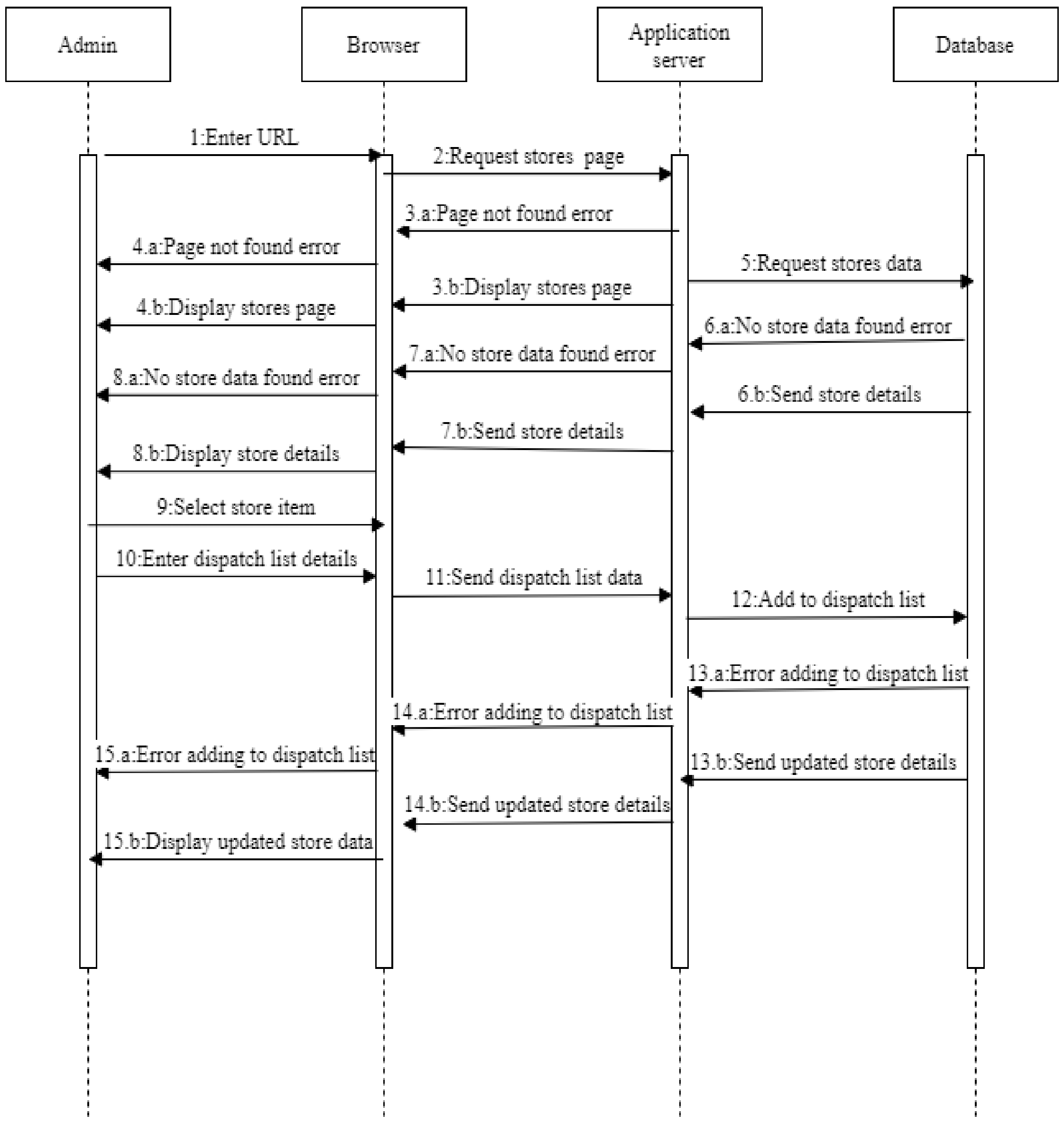
10)Add item to production sequence diagram


Admin
Browser
Application server
Database
1:Enter URL
2:Request pending production page
3.a:Page not found error
4.a:Page not found error
5:Request pending production data
3.b:Display pending production page
4.b:Display pending production page
6.a:No items found error
7.a:No items found error
8.a:No items found error
6.b:Send pending production details
7.b:Send pending production details
8.b:Display pending production details
9:Select item
10:Enter quantity to produce
11:Send item data
12:Update pending production data
13.a:Error updating record
14.a:Error updating record
15.a:Error updating record
13.b:Send pending production details
14.b:Send pending production details
15.b:Display updated data

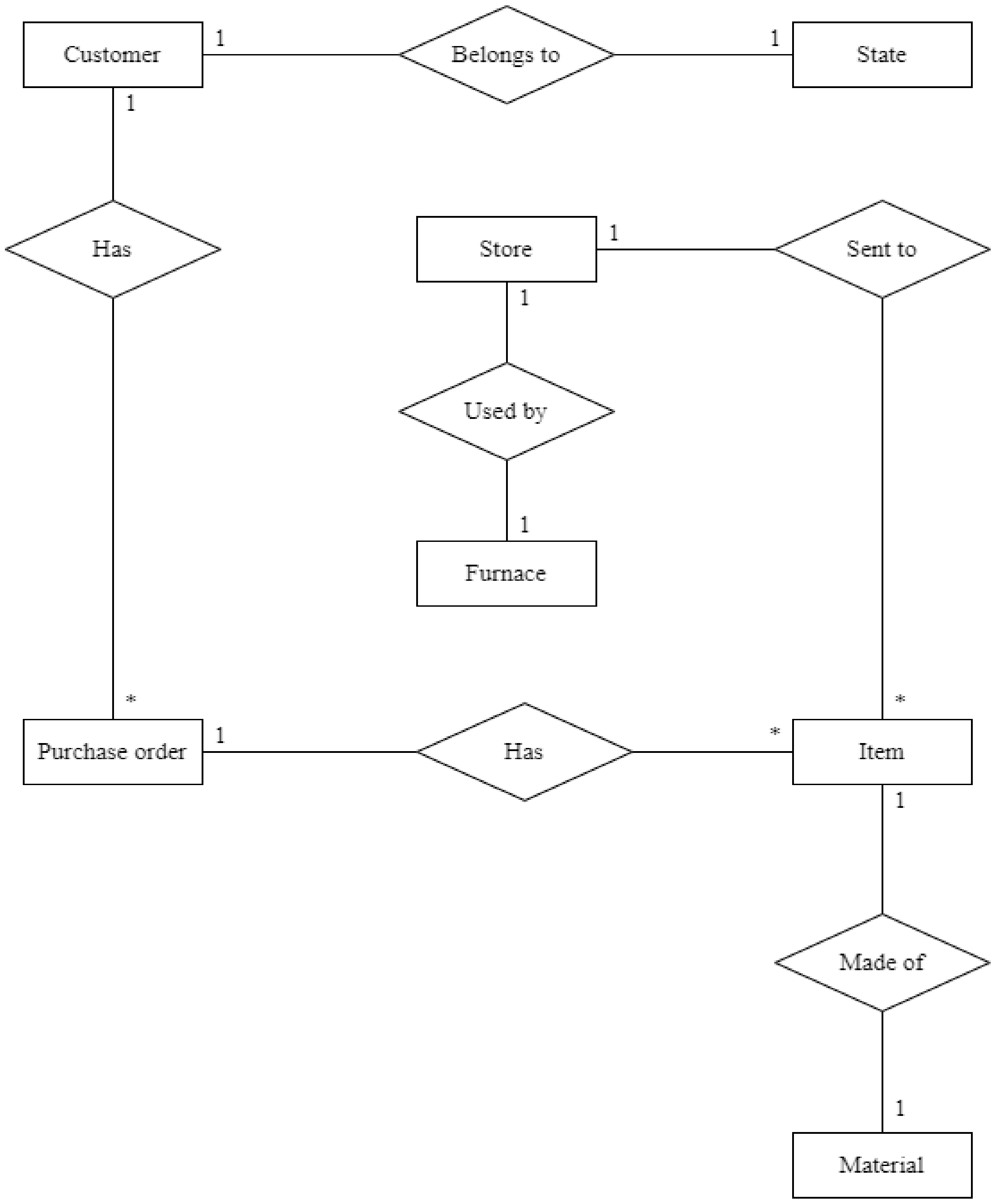
11)Send to store sequence diagram


Admin
Browser
Application server
Database
1:Enter URL
2:Request in-production page
3.a:Page not found error
4.a:Page not found error
5:Request in-production data
3.b:Display in-production page
4.b:Display in-production page
6.a:No items found error
7.a:No items found error
8.a:No items found error
6.b:Send in-production details
7.b:Send in-production details
8.b:Display in-production details
9:Select item
10:Enter produced item details
11:Send produced item data
12:Add to store
13.a:Error adding to store
14.a:Error adding to store
15.a:Error adding to store
13.b:Send updated in-production details
14.b:Send updated in-production details
15.b:Display updated in-production data

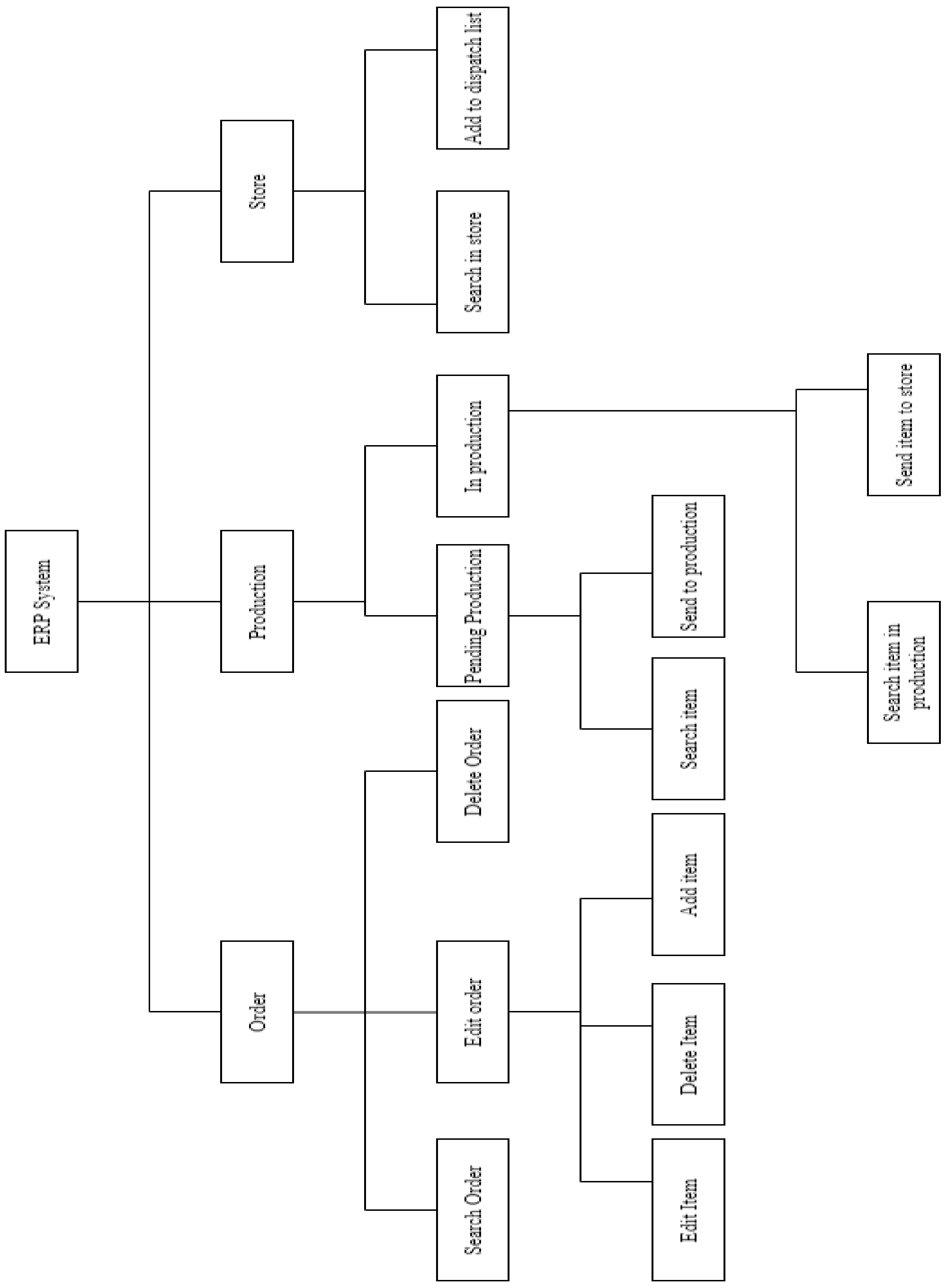
12)Add to dispatch list sequence diagram


Admin
Browser
Application server
Database
1:Enter URL
2:Request stores page
3.a:Page not found error
4.a:Page not found error
5:Request stores data
3.b:Display stores page
4.b:Display stores page
6.a:No store data found error
7.a:No store data found error
8.a:No store data found error
6.b:Send store details
7.b:Send store details
8.b:Display store details
9:Select store item
10:Enter dispatch list details
11:Send dispatch list data
12:Add to dispatch list
13.a:Error adding to dispatch list
14.a:Error adding to dispatch list
15.a:Error adding to dispatch list
13.b:Send updated store details
14.b:Send updated store details
15.b:Display updated store data

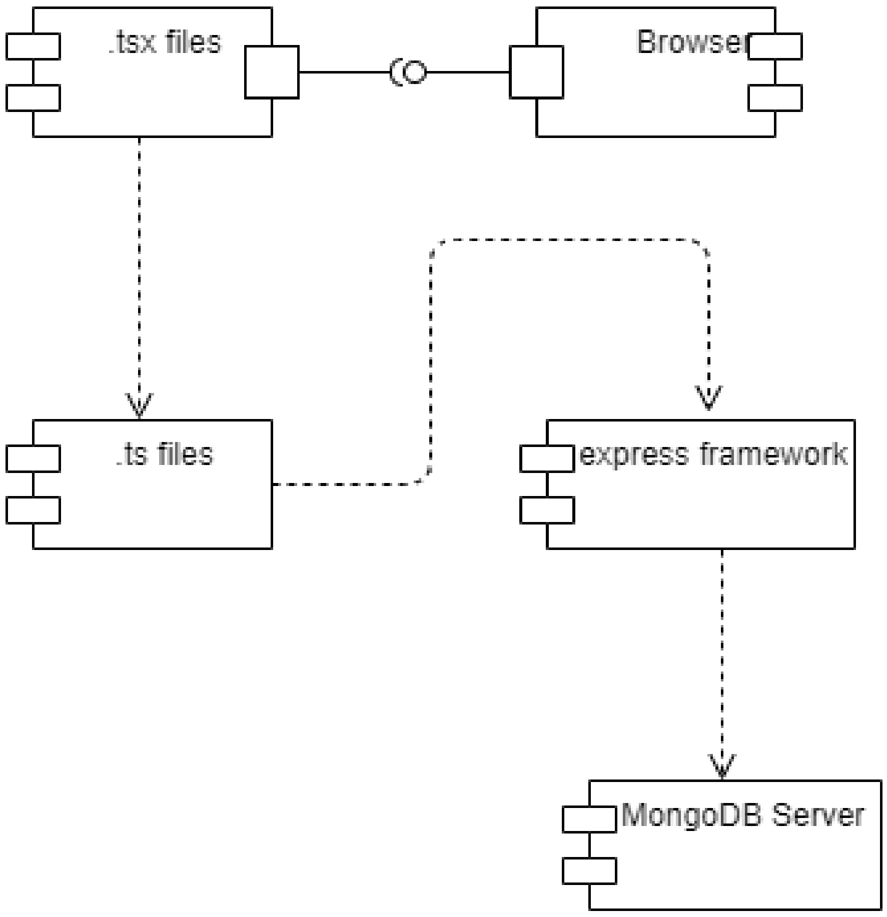
3.6 ERD


Customer
1
Belongs to
1
State
1
Has
Store
1
Sent to
1
Used by
1
Furnace
*
Purchase order
1
Has
*
Item
*
1
Made of
1
Material

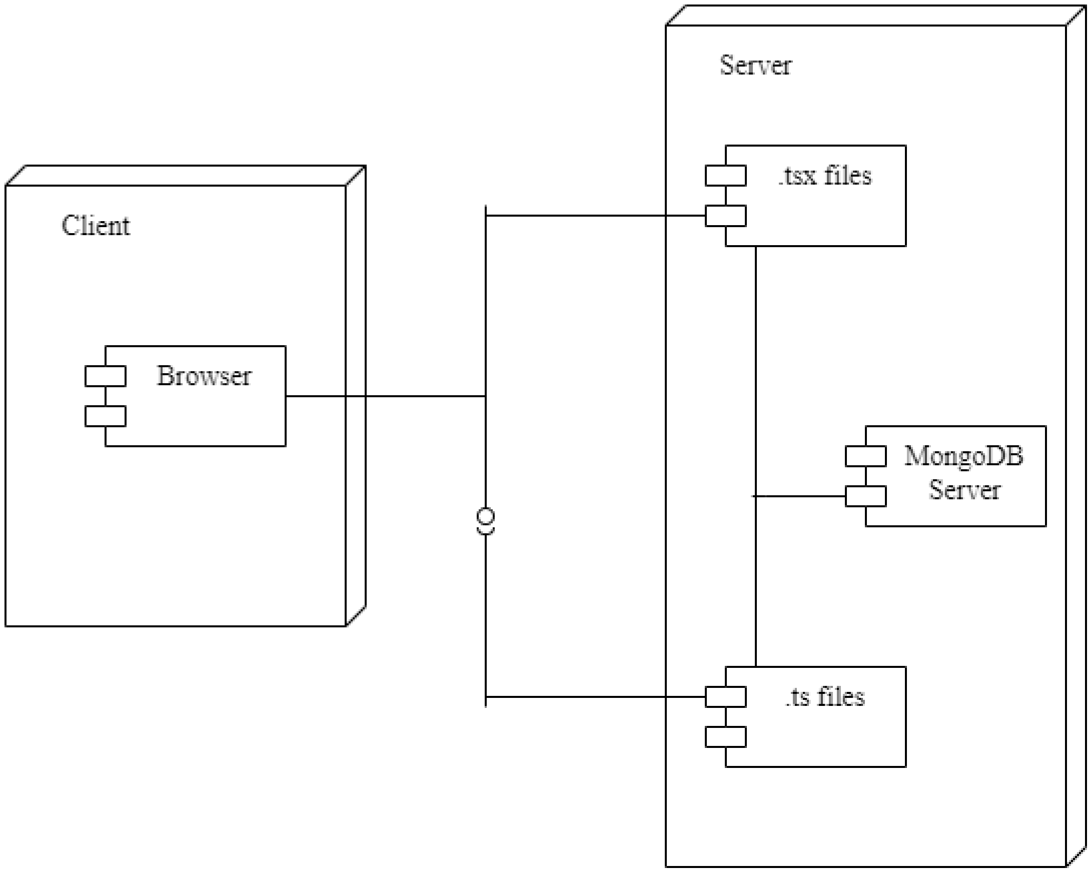
3.7 Module Hierarchy diagram



3.8 Component Diagram



3.9 Deployment Diagram



3.10 Module Specifications

As per the module hierarchy diagram there are 3 main modules in the project

- Order
- Production
- Store

Order

This module displays the order data. User can search for particular order using specific filters. In addition to this user is able to delete and edit and order

If the user wants to edit the order a new page loads with fields already populated with order data. The items included in the order are shown in a table where clicking on the edit item button opens a pop up form populated with that item details where user can change the content. In addition to this a new item can also be added in the order by clicking on add new item button which opens a pop up form to collect data for the new item.

Production

This module displays 2 tabs. One tab shows items which are yet to be produced. User can search for particular item by using specific filters. User can select the quantity of items to be sent into production

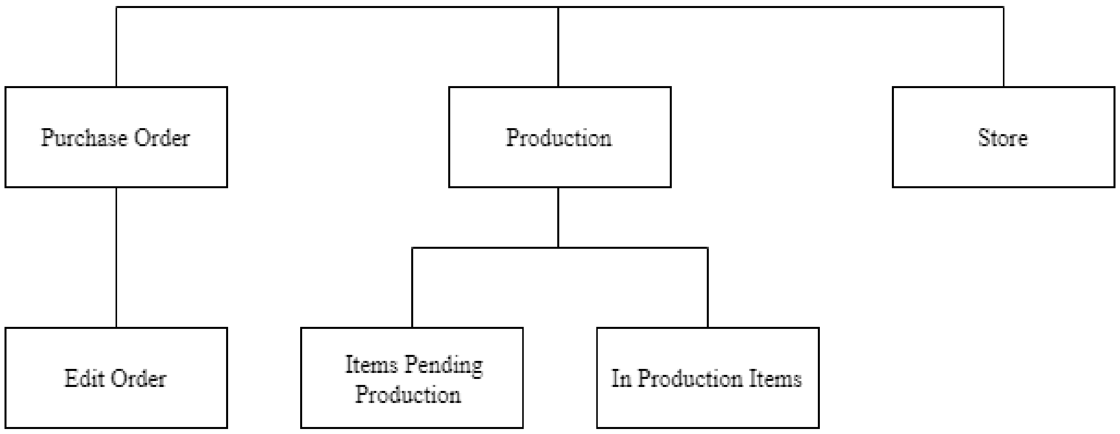
The other tab shows the items which are in production. Similar kind of filters are available to search items in production. User sends finished item to the store by clicking the send to store button which opens a pop up form to collect information about the finished product.

Store

This module displays the finished products of an order and displays the details on a table. These products can be searched using specific filters.

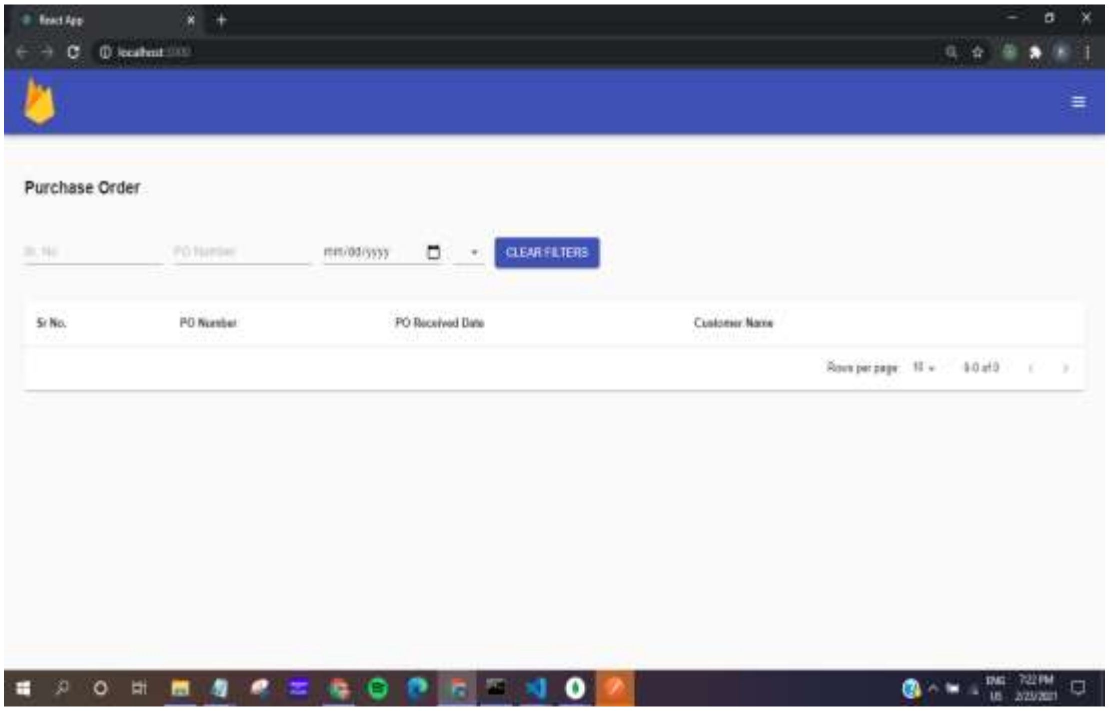
User is able to add products to dispatch by clicking add to dispatch button and entering quantity of products to dispatch.

3.11 Website Map

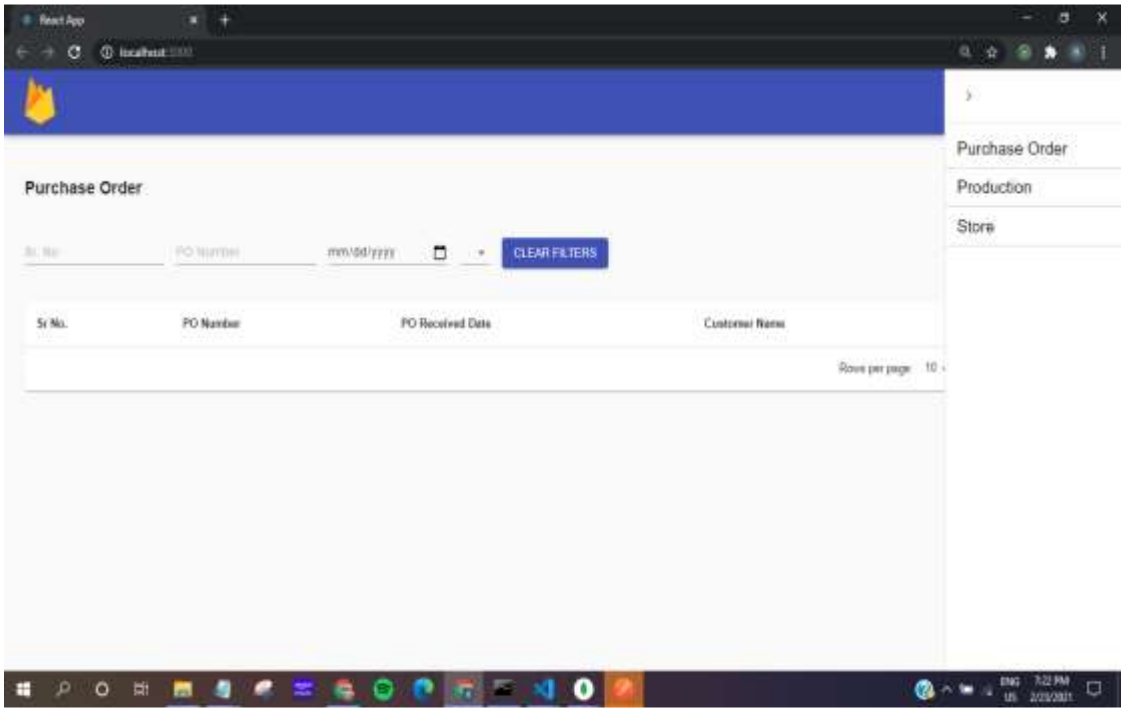


3.12 User Interface Design

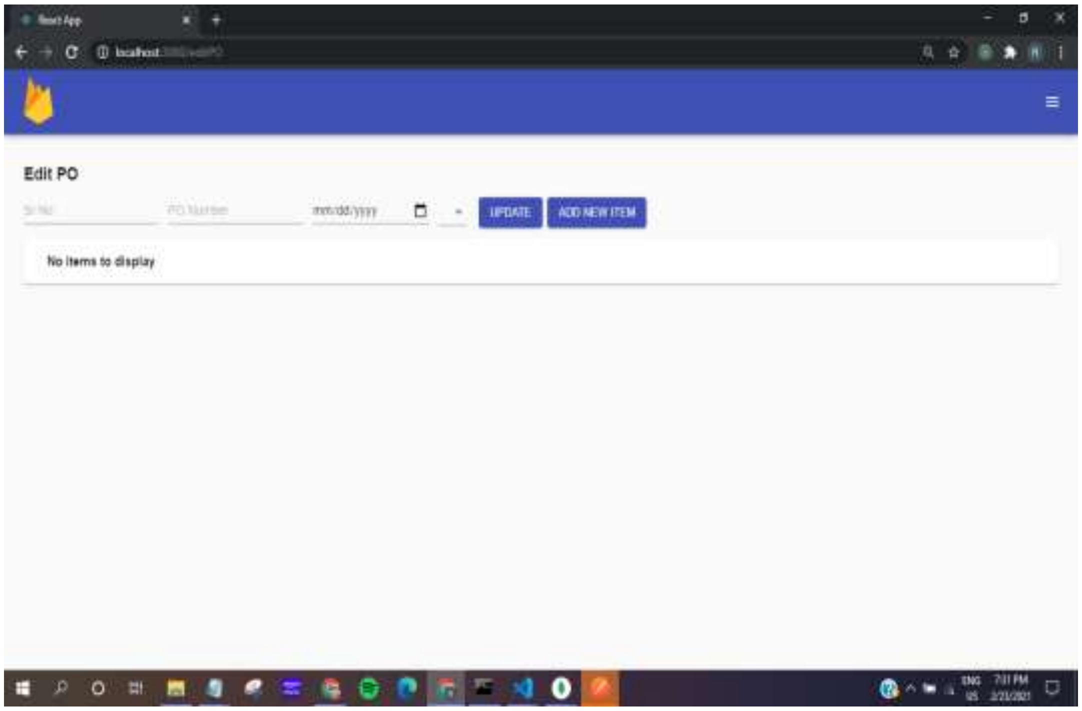
Purchase Order



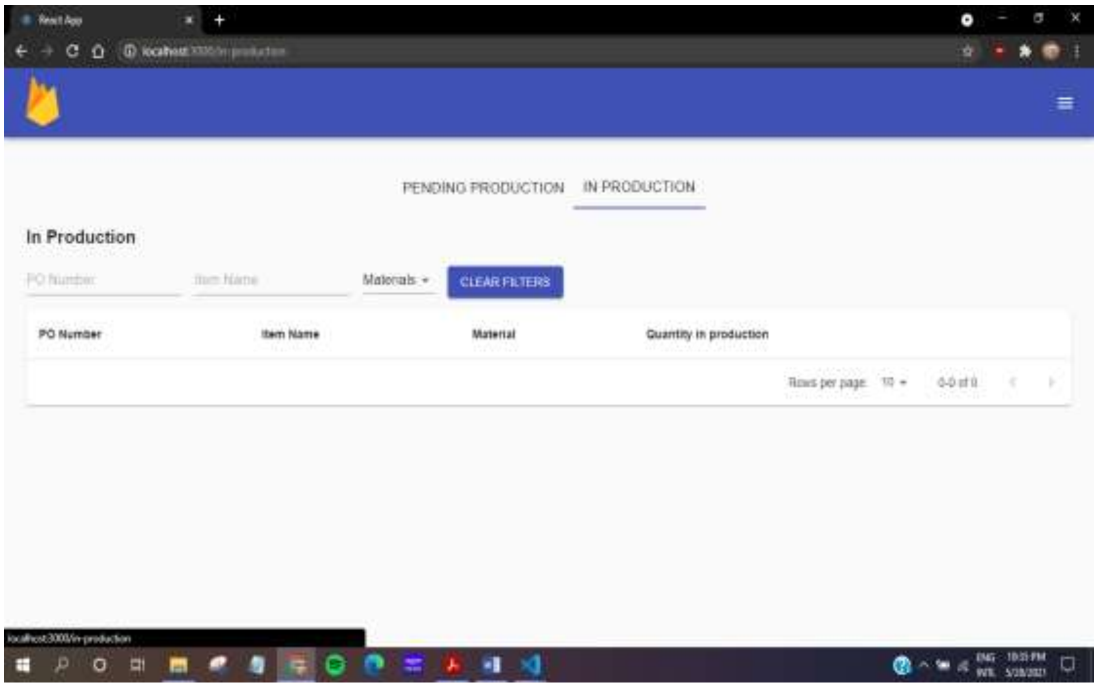
Navigation Bar



Edit Order


React App
localhost
Edit PO
Sr No
PO Number
mm/dd/yyyy
UPDATE
ADD NEW ITEM
No items to display

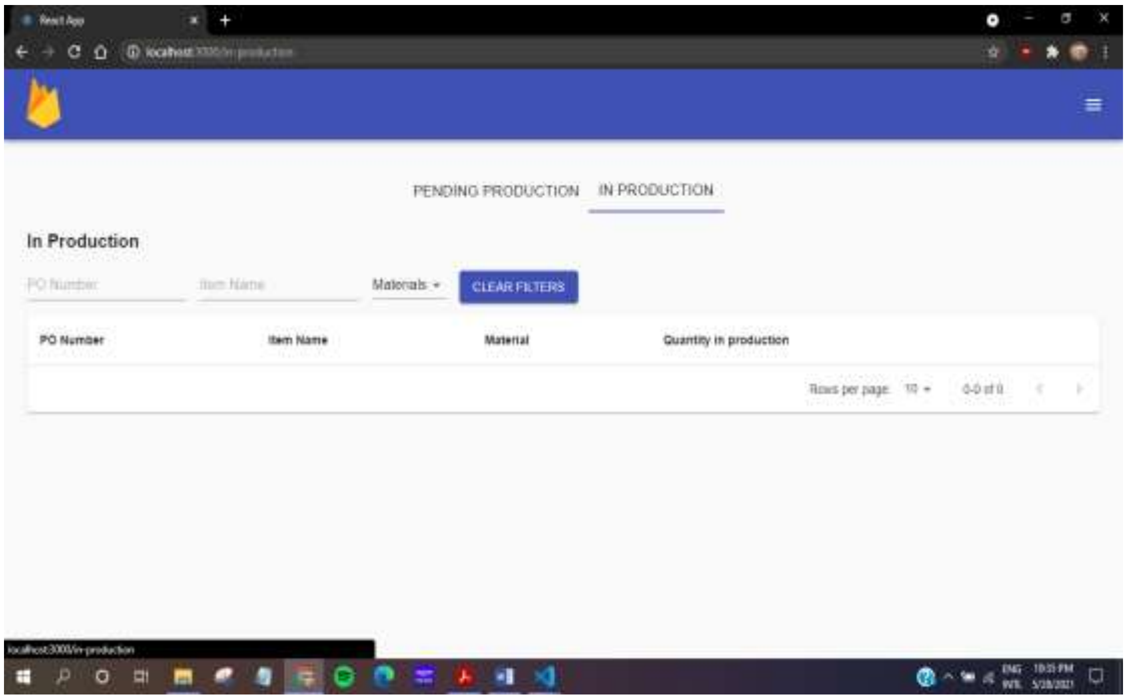
In Production Items



In Production

PO Number	Item Name	Material	Quantity in production

Items pending production

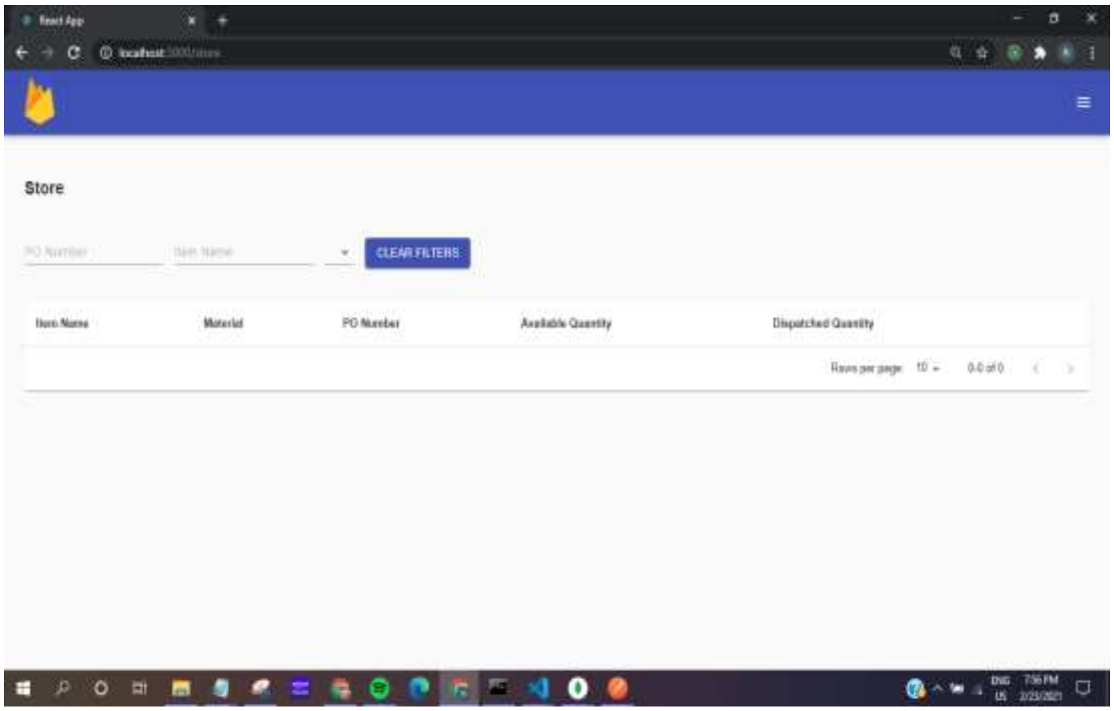


In Production

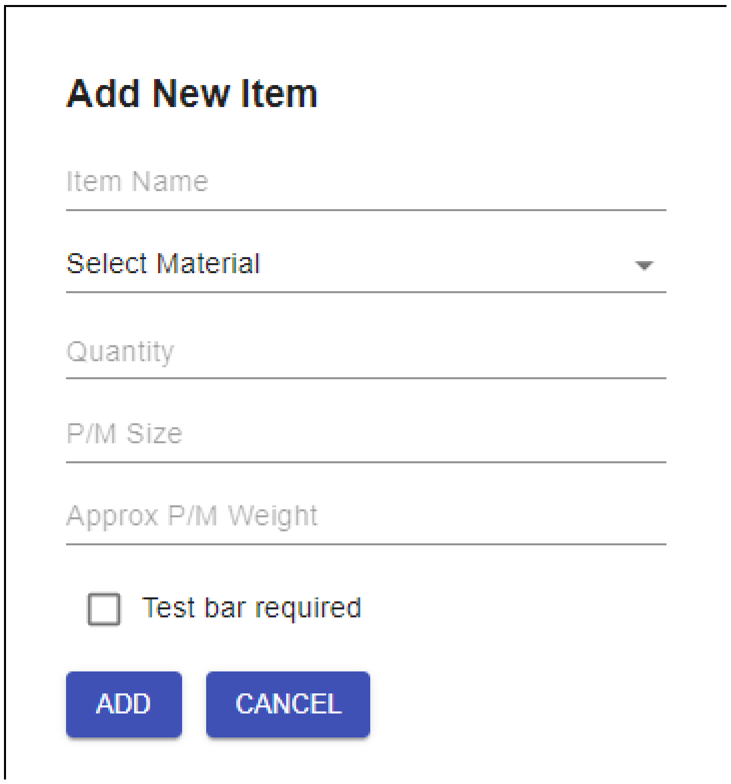
PO Number	Item Name	Material	Quantity in production

Rows per page: 10 0-0 of 0

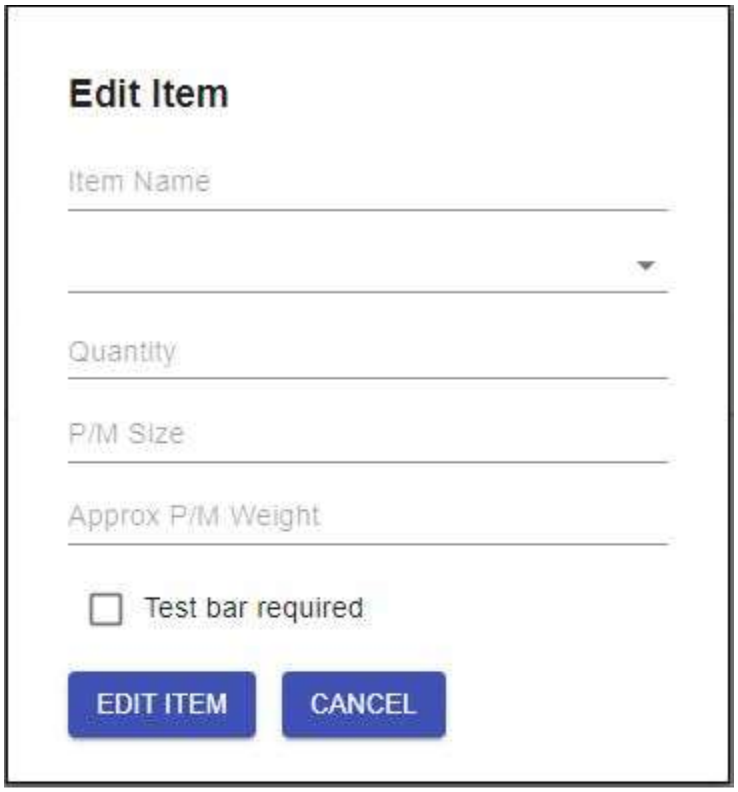
Store


React App
localhost
Store
PO Number
Item Name
CLEAR FILTERS
Item Name
Material
PO Number
Available Quantity
Dispatched Quantity
Rows per page: 10
0-0 of 0

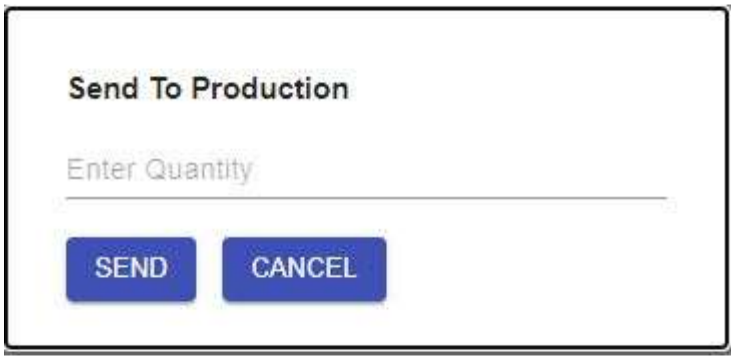
Add item pop-up



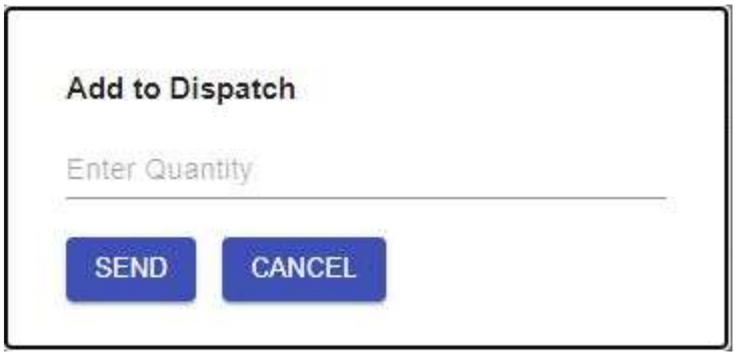
Edit Item pop-up



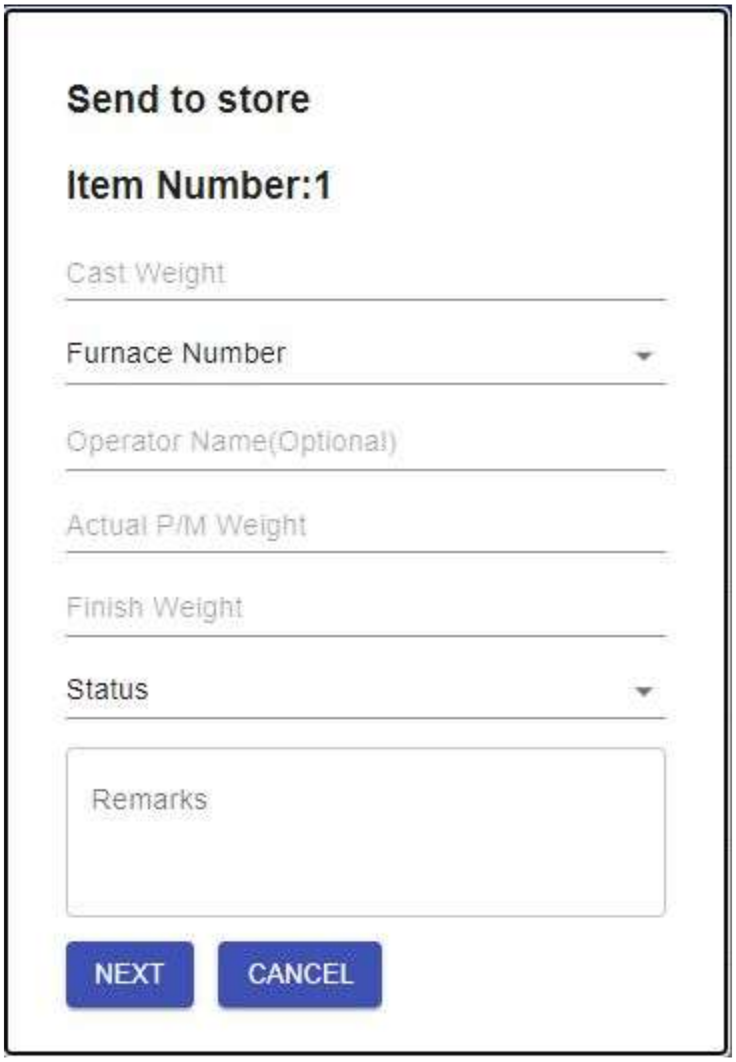
Send to production pop-up



Add to dispatch pop-up



Send to store pop-up


Send to store

Item Number:1

Cast Weight

Furnace Number

Operator Name(Optional)

Actual P/M Weight

Finish Weight

Status

Remarks

3.14 Data Dictionary

SrNo	Field	Data Type	Description
1	abbr	String(2)	State short form
2	actualPmWeight	Number(3)	Actual Weight
3	address	String(50)	Address of customer
4	approxPmWeight	Number(3)	Weight of item ordered
5	castWeight	Number(3)	Cast weight of item
6	customerID	ObjectID	Primary key for customer,Foreign key for purchase order table
7	customerName	String(25)	Name of customer
8	customerPan	String(10)	PAN of customer
9	finishWeight	Number(3)	Finished weight
10	furnaceID	ObjectID	Primary key in furnace table,foreign key in store table
11	furnaceNumber	String(10)	Furnace number
12	GSTNo	String(15)	GST Number of customer

13	isCustomerDeleted	boolean	Indicator to show if record is deleted
14	isDispatched	Boolean	Indicator that item is in dispatch list or not
15	isItemDeleted	Boolean	Indicator to show if item is deleted
16	isPoDeleted	Boolean	Indicator to show if record is deleted
17	itemID	ObjectID	Item primary key,foreign key in stores table
18	itemName	String(25)	Name of item
19	itemsList	Array	Array of ordered item ObjectIDs
20	materialID	ObjectID	Primary key of material table, foreign key in Item table
21	materialName	String(15)	Name of material
22	operatorName	String(15)	Name of operator
23	pendingQuantity	Number(3)	Items not in production

24	pmSize	String(10)	Size of item ordered
25	poID	ObjectID	Primary key for purchase order table,foreign key in Item table,stores table
26	poNumber	String(15)	Purchase order number
27	poReceivedDate	Date(10)	Date order is received
28	producedItems	Array	Array of stored item ObjectIDs
29	quantity	Number(3)	Quantity of items ordered
30	quantityAvailable	Number(3)	Number of items available
31	quantityDispatch	Number(3)	Number of items dispatched
32	quantityInProduction	Number(3)	Items in production
33	remarks	String(25)	Comments about stored item

34	srno	String(15)	Serial number
35	stateID	ObjectID	Primary key of state table,foreign key in purchase order table
36	stateName	String(20)	Name of state
37	status	String(10)	Status of produced item
38	storeID	ObjectID	Primary key in store table
39	vendorCode	String(10)	Vendor code of customer

3.15 Table Specifications

1)Purchase-Orders

Field	Data type	Width	Constraint
poId	ObjectID		Primary key
srno	String	15	Not null
poNumber	String	15	Not null
poReceivedDate	Date	10	Not null
itemsList	Array		Not null
customerID	ObjectID		Foreign Key,Not null
isPoDeleted	Boolean	1	Not null

2)Customers

Field	Data type	Width	Constraint
customerID	ObjectID		Primary key
customerName	String	15	Not null
address	String	50	Not null
customerPan	String	10	Not null
vendorCode	String	10	Not null
GSTNo	String	15	Not null
isCustomerDeleted	Boolean	1	Not null
stateID	ObjectID		Foreign key

3)State

Field	Data type	Width	Constraint
stateID	ObjectID		Primary key
stateName	String	20	Not Null
abbr	String	2	Not Null

4)Items

Field	Data type	Width	Constraint
itemID	ObjectID	ObjectID	Primary Key
itemName	String	25	Not null
Quantity	Number	3	Not null
pmSize	String	10	Not null
approxPmWeight	Number	3	Not null
pendingQuantity	Number	3	Not null
quantityInProduction	Number	3	Not null
quantityAvailable	Number	3	Not null
quantityDispatch	Number	3	Not null
producedItems	Array		
isItemDeleted	Boolean	1	Not null
materialID	ObjectID		Foreign Key
poID	ObjectID		Foreign Key

5)Material

Field	Data type	Width	Constraint
materialID	ObjectID		Primary Key
materialName	String	15	Not null

6)Store

Field	Data type	Width	Constraint
storeID	ObjectID		Primary key
operatorName	String	15	
remark	String	25	
castWeight	Number	3	Not null
actualPmWeight	Number	3	Not null
finishWeight	Number	3	Not null
Status	String	10	Not null
isDispatched	Boolean	1	Not null
furnaceID	ObjectID		Foreign Key
poID	ObjectID		Foreign Key
itemID	ObjectID		Foreign Key

7)Furnace

Field	Data type	Width	Constraint
furnaceID	ObjectID		Primary Key
furnaceNumber	String	10	Not null

3.16 Test procedures and implementation

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation. It is a process of executing a program with a primary objective of finding errors. Testing gives the guarantee that the software does not fail and runs according to its specifications and in the way the end user expects.

Testing will be performed by running the program using the test data. Testing is vital to the success of the system. It will also test whether the system identify the problem correctly.
The following software testing techniques were used in order to uncover errors in the system:

- Unit testing
- Integration testing

Unit Testing

Unit testing is normally considered as an adjunct to the coding step. It is the test for the small units of code, e.g. programs, modules or procedures, in order to ensure that they perform their intended functions. Unit testing is also done to test the data flow across a module interface.

The following errors are uncovered during unit testing:

- Comparison of different data types.
- Incorrect logical operators or precedence.
- Incorrect comparison of variables.
- Improper or nonexistent loop termination.
- Improperly modified loop variable.

Integration testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. During this activity, unit tested components are taken and a program structure is built as per the design. Then incremental integration is performed on the system. This means that programs are constructed and tested in small increments instead of testing the entire program as a whole. This is done because correction of errors becomes difficult in case of whole program testing as many errors were detected and it is not easy to correct them at one go. Thus, through incremental integration testing, any error uncovered could be easily noted and corrected and interfaces are tested completely.

Test Cases for add new item and edit item

Test No.	Description	Input	Expected Results	Actual Results	Pass/ Fail
1	Item name should not be null	itemName=NULL	Item Name is required	Item Name is required	Pass
2	A material should be selected	materialName=NULL	Material is required	Material is required	Pass
3	Quantity should not be null	quantity=NULL	Quantity is required	Quantity is required	Pass
4	PM Size should not be null	pmSize=NULL	PM Size is required	PM Size is required	Pass

5	Approx. PM Weight should not be null	approxPmWeight= NULL	Approx. PM Weight is required	Approx. PM Weight is require d	Pass
6	Item name should be string	itemName='pipe'	No Error	No Error	Pass
7	Material is selected	materialName='Ob jectID'	No error	No error	Pass
8	Quantity should be integer	Quantity=8	No error	No error	Pass
9	PM Size should be string	pmSize='small'	No error	No Error	Pass
10	Approx. PM Weight should be float	approxPmWeight= 25.5	No Error	No Error	Pass

Test Cases for send to store pop-up

Test No.	Description	Input	Expected Results	Actual Results	Pass/ Fail
1	Cast Weight should not be null	castWeight=NULL	Cast Weight is required	Cast Weight is required	Pass
2	A furnace number should be selected	furnaceNumber=NULL	Furnace number is required	Furnace number is required	Pass
3	Operator name can be null	operatorName =NULL	No Error	No Error	Pass
4	Remarks can be null	Remarks=NULL	No Error	No Error	Pass
5	Finish Weight should not be null	finishWeight= NULL	Finish Weight is required	Finish Weight is required	Pass
6	Status should be selected	Status=NULL	Select status	Select status	Pass

7	Cast Weight should be float	castWeight=25.5	No Error	No Error	Pass
8	Furnace number is selected	furnaceNumber=’F1’	No error	No error	Pass
9	Finish Weight should be float	finishWeight=25.5	No error	No Error	Pass
10	Status should be selected	status=’OK’	No Error	No Error	Pass

Test Cases for Edit Purchase order

Test No.	Description	Input	Expected Results	Actual Results	Pass/Fail
1	srNo should not be null	srNo=NULL	srNo is required	srNo is required	Pass
2	poID should not be null	poID=NULL	poID is required	poID is required	Pass
3	srNo should not be null	srNo=”SC101”	No error	No error	Pass
4	poID should not be null	poID=”SCPOLS1”	No error	No error	Pass

Test Cases for Send to Production

Test No.	Description	Input	Expected Results	Actual Results	Pass/Fail
1	Enter quantity of items to be sent to production	quantity=NULL	Quantity should be more than 0	Quantity should be more than 0	Pass
2	Enter quantity of items to be sent to production	quantity=2	No error	No error	Pass

Test Cases for Add to dispatch

Test No.	Description	Input	Expected Results	Actual Results	Pass/Fail
1	Enter quantity of items to be sent to dispatch	quantity=NULL	Quantity should be more than 0	Quantity should be more than 0	Pass
2	Enter quantity of items to be sent to dispatch	quantity=2	No error	No error	Pass

Integration Tests

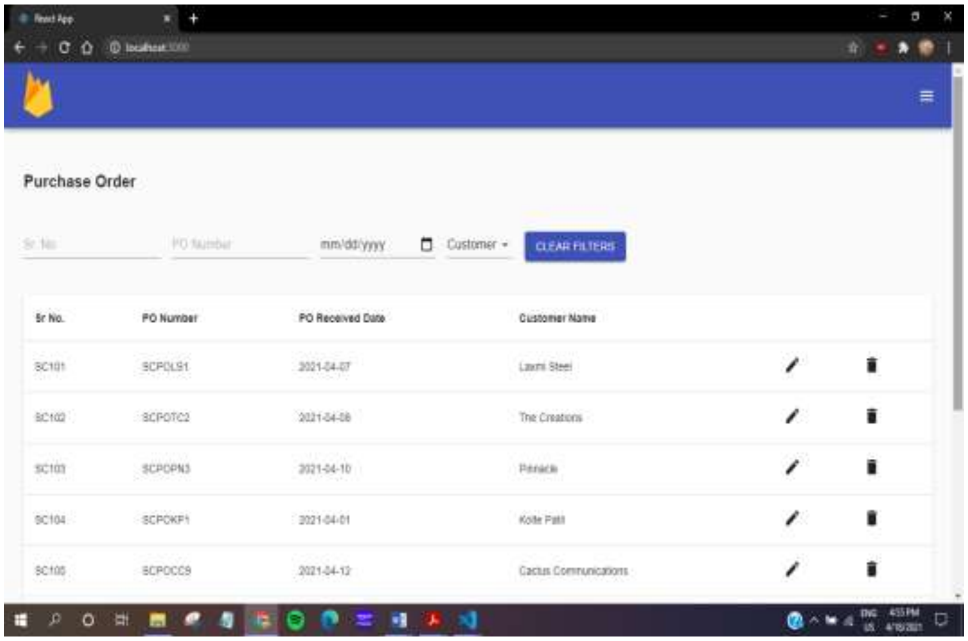
Test No.	Description	Input	Expected Results	Actual Results	Pass/Fail
1	Open Edit Order Page from Purchase orders page	Click edit button of a table row	Fields and item table displayed populated with row data	Fields and item table displayed populated with row data	Pass
2	Moving from one page to another	Click a nav bar button	Respective page is loaded	Respective page is loaded	Pass

Chapter 4 – User Manual

4.1 User Manual

User manual is document provided for the user to see how computerized system works actually. It describes everything about how the system can be used i.e. how data is to be entered in to the controls.

Purchase Order Page



Purchase Order

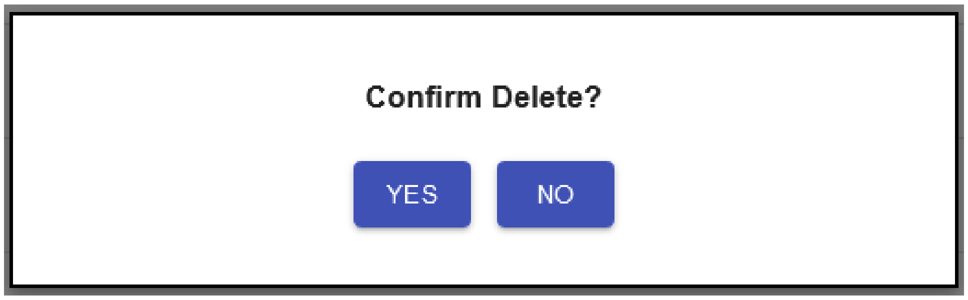
Sr No.	PO Number	PO Received Date	Customer Name
SC101	SCPOLS1	2021-04-07	Laxmi Steel
SC102	SCPOTC2	2021-04-08	The Creations
SC103	SCPOPN3	2021-04-10	Pinnacle
SC104	SCPOKP1	2021-04-01	Kolte Patil
SC105	SCPOCC9	2021-04-12	Cactus Communications

This is the first page that loads as the web app runs. User is also able to navigate to this page using the sidebar menu. This page has a table that displays the orders that the company has received. The table displays a serial number, purchase order number, date on which an order is received and the name of the customer who has placed the order.

The table data can be filtered using the fields above the table. As a user enters data in any of fields, data is reloaded in the table according to the results returned from the backend. The “CLEAR FILTERS” button clears the filtering parameters and reloads the table as it was before applying the filters

The table has an edit button and a delete button for every row.

Clicking on the delete button of a row a pop-up appears asking to confirm the deletion of the record.

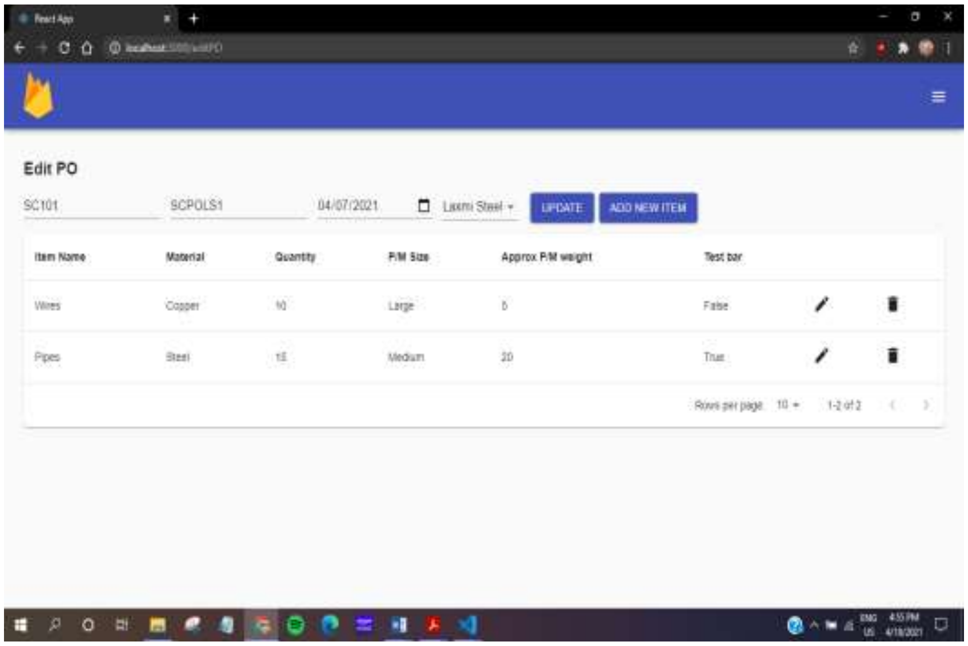


If clicked on “YES” button, the record is deleted and the table data reloads.

If “NO” button is clicked the pop-up is closed.

Clicking on edit button redirects the application on the “Edit Purchase Order” page.

Edit Purchase Order Page



Edit PO

SC101 | SCPOLS1 | 04/07/2021 | Laxmi Steel

Item Name	Material	Quantity	P/M Size	Approx P/M weight	Test bar
Wires	Copper	10	Large	5	False
Pipes	Steel	15	Medium	20	True

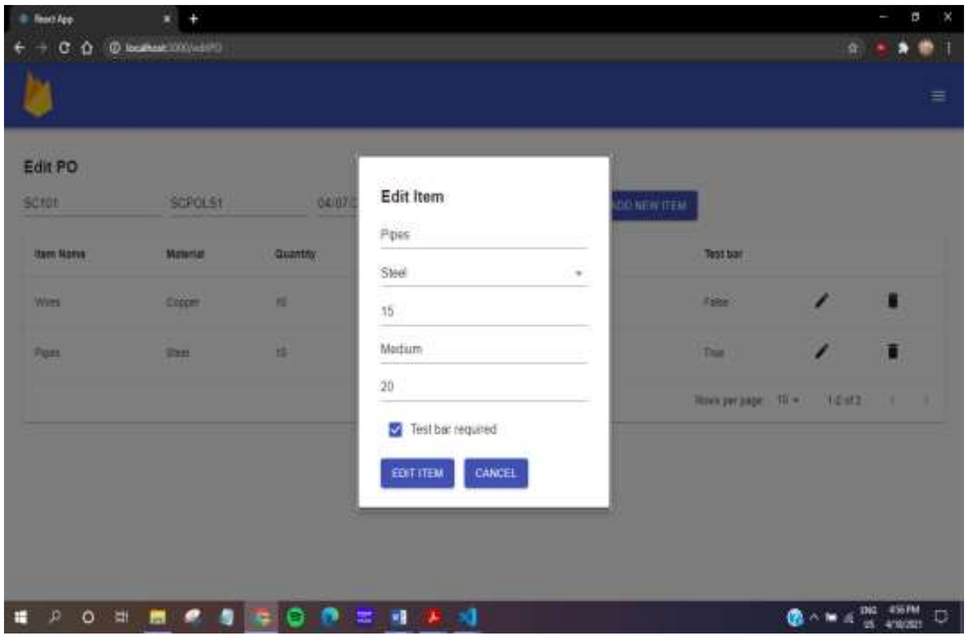
Rows per page: 10 1-2 of 2

User is able to navigate to this page only from the "Purchase Order" page by clicking on the edit button in any row from the table. When the page loads, order details are already populated in their respective fields. User can change the data in any of these fields. Clicking on "UPDATE" button updates the order data and the user is again redirected to the "Purchase Order" page.

The details of items in the order are populated in a table below the fields. The table has an edit button and a delete button for every row. Clicking on the delete button of a row a pop-up appears asking to confirm the deletion of the record. If clicked on "YES" button, the record is deleted and the table data reloads.

If "NO" button is clicked the pop-up is closed.

If edit button is clicked a pop-up appears with fields populated with the data of the selected row. Clicking on "EDIT ITEM" updates the item data and table data is reloaded.



Edit PO

SC101 SCPOL51 04/07/

Item Name	Material	Quantity	Test bar
Wires	Copper	15	False
Pipes	Steel	15	True

Edit Item

Pipes

Steel

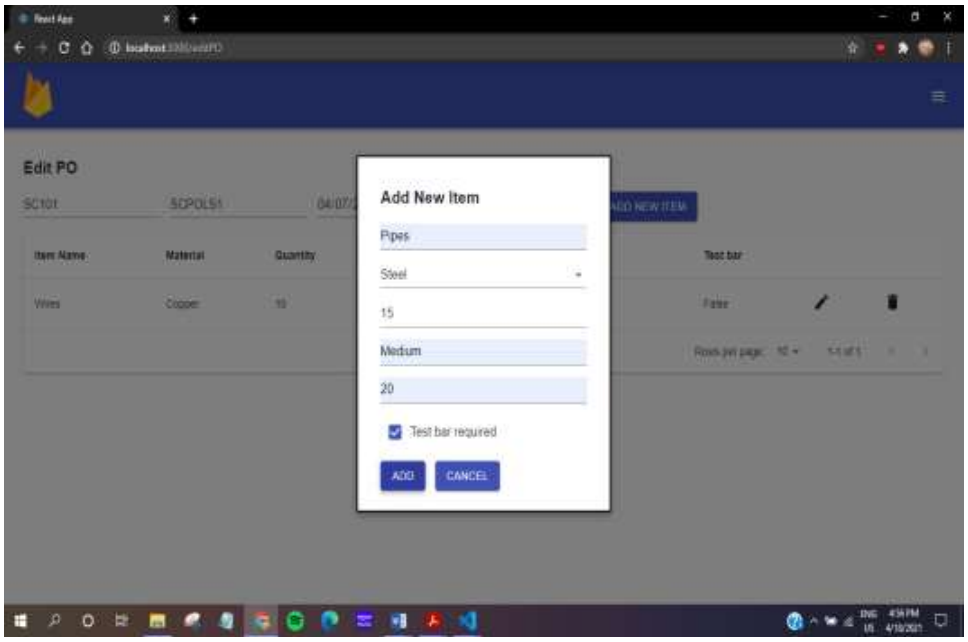
15

Medium

20

☑ Test bar required

User can also add an item in the order. By clicking on "ADD NEW ITEM" button a pop-up appears where the user can input the new item data. By clicking on the "ADD" button, the item is added in the order and the table is reloaded to show the updated items list.



SC101 SCPOLS1 04/07/

Item Name	Material	Quantity	Test bar
Wires	Copper	10	False

Pipes

Steel

15

Medium

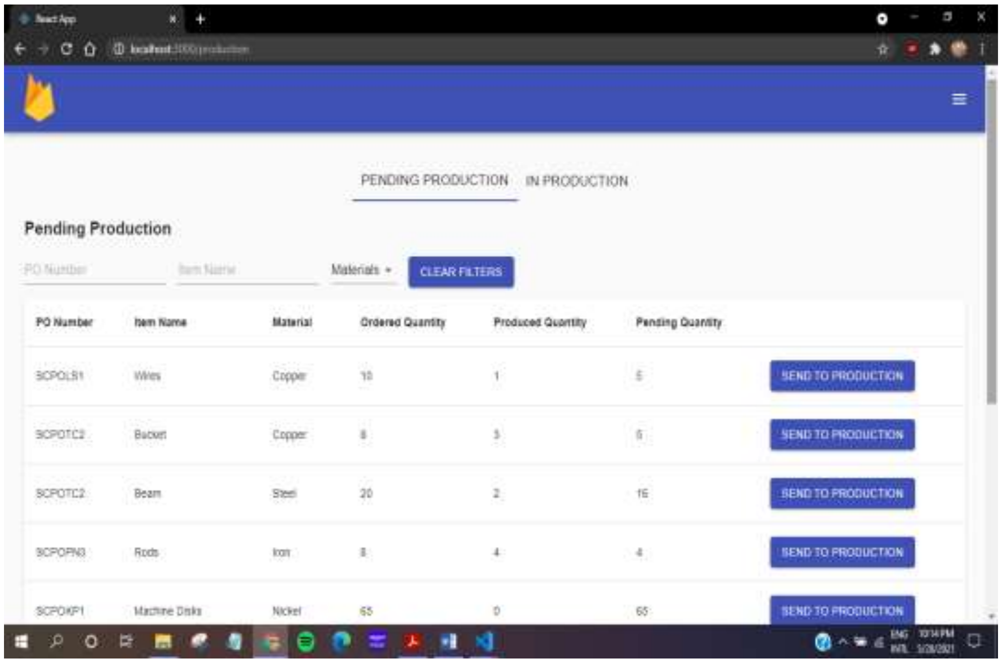
20

[✓] Test bar required

Production Page

This page is used to monitor and manage the production of the items of an order. User is able to navigate to this page from the sidebar. This page has 2 tabs –

1)Pending Production –



Pending Production

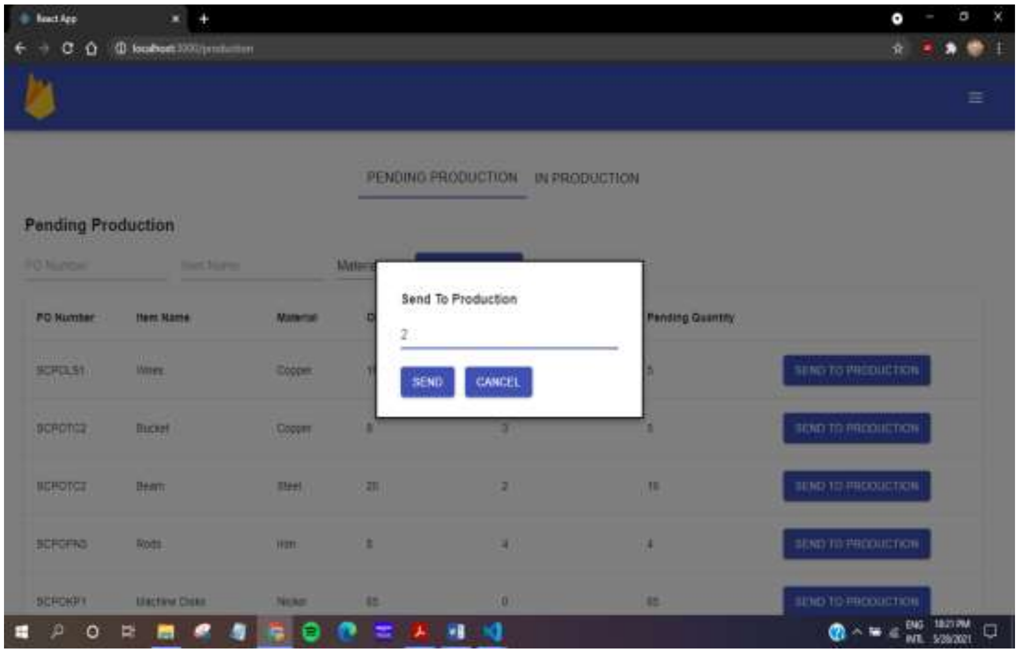
PO Number

Item Name

PO Number	Item Name	Material	Ordered Quantity	Produced Quantity	Pending Quantity	
SCPOLS1	Wires	Copper	10	1	5	SEND TO PRODUCTION
SCPOTC2	Bucket	Copper	8	3	5	SEND TO PRODUCTION
SCPOTC2	Beam	Steel	20	2	15	SEND TO PRODUCTION
SCPOPN3	Rods	Iron	8	4	4	SEND TO PRODUCTION
SCPOKP1	Machine Disks	Nickel	65	0	65	SEND TO PRODUCTION

When the production page loads, the "PENDING PRODUCTION" tab is displayed by default. This tab has a table showing the order items and their details such as item name, material, ordered quantity, produced quantity, and pending quantity.

The table data can be filtered using the fields above the table. As a user enters data in any of fields, data is reloaded in the table according to the results returned from the backend. The “CLEAR FILTERS” button clears the filtering parameters and reloads the table as it was before applying the filters.



Pending Production

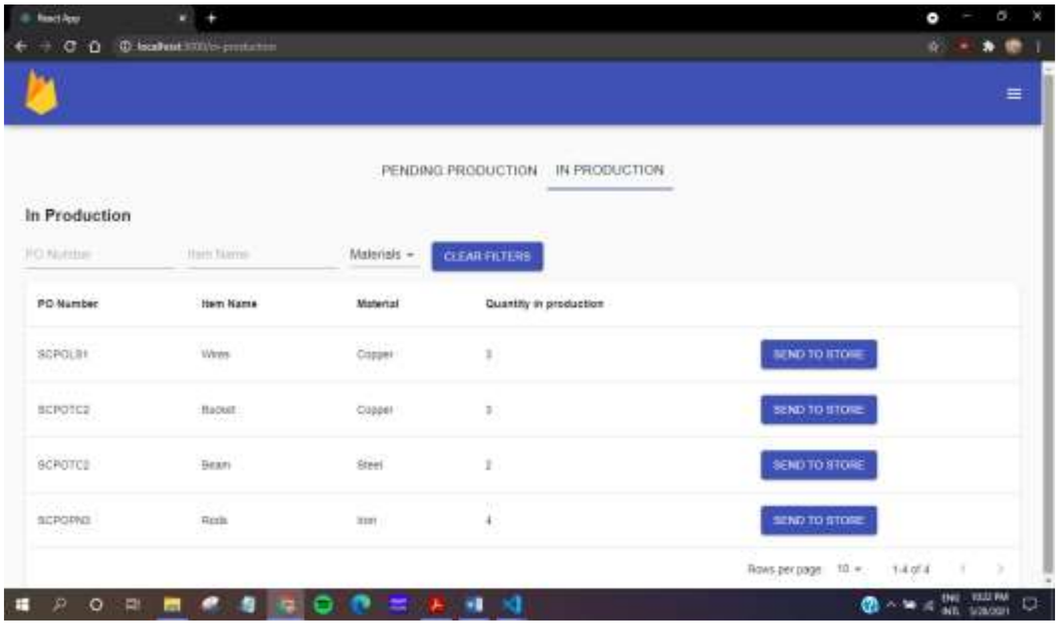
Send To Production

2

PO Number	Item Name	Material	[illegible]	[illegible]	Pending Quantity	
SCPOLS1	Wires	Copper	[illegible]	[illegible]	5	SEND TO PRODUCTION
SCPOTC2	Bucket	Copper	8	3	5	SEND TO PRODUCTION
SCPOTC2	Beam	Steel	20	2	18	SEND TO PRODUCTION
SCPOFN3	Rods	Iron	8	4	4	SEND TO PRODUCTION
SCPOKP1	Machine Disks	Nickel	85	0	85	SEND TO PRODUCTION

Each row in the table has a “SEND TO PRODUCTION” button. When it is clicked a pop-up appears with one field to enter quantity of items to send to production. Clicking on the “SEND” button updates the item data and reloads the table.

2)In Production –



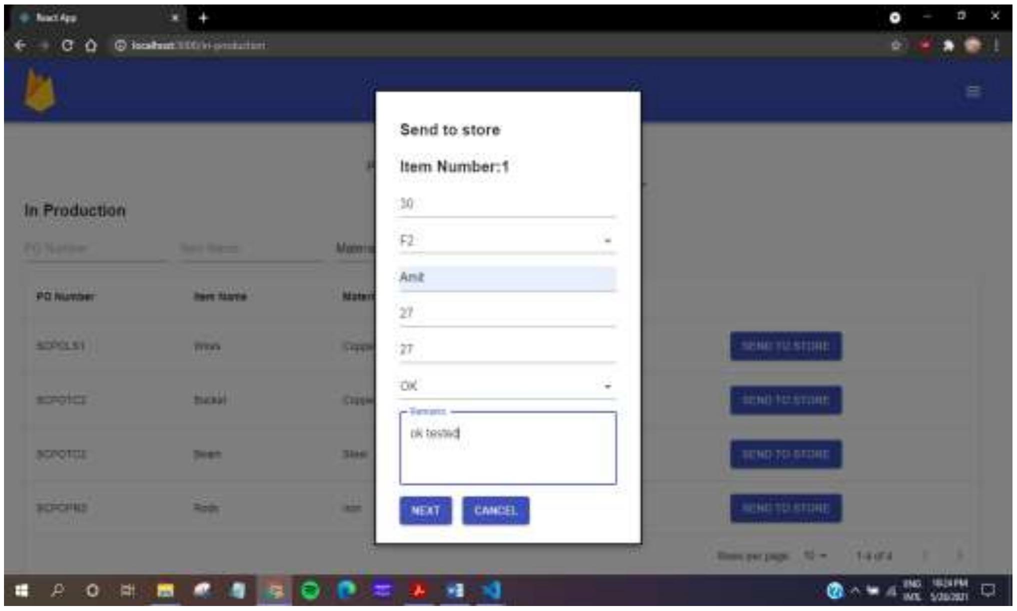
In Production

PO Number	Item Name	Material	Quantity in production	
SCPOLB1	Wires	Copper	3	SEND TO STORE
SCPOTC2	Racket	Copper	3	SEND TO STORE
SCPOTC2	Beam	Steel	2	SEND TO STORE
SCPOPN3	Rods	Iron	4	SEND TO STORE

Rows per page 10 1-4 of 4

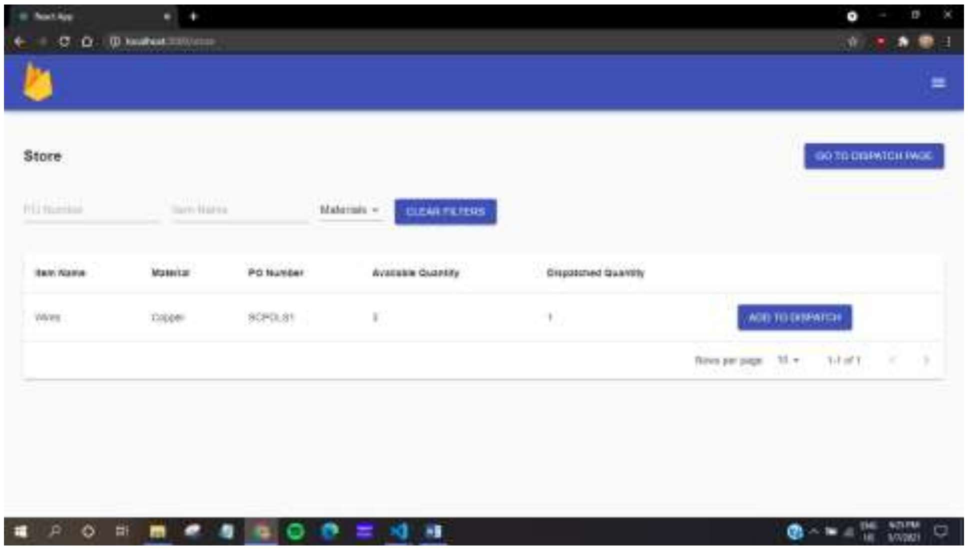
This tab opens after clicking on "IN PRODUCTION" tab button. This tab has a table showing the order items that are in production and their details such as item name, material and quantity in production.

The table data can be filtered using the fields above the table. As a user enters data in any of fields, data is reloaded in the table according to the results returned from the backend. The "CLEAR FILTERS" button clears the filtering parameters and reloads the table as it was before applying the filters.



Each row in the table has a “SEND TO STORE” button. When it is clicked a pop-up appears with input fields for information about a finished item. Clicking on the “SEND” button sends the item to the store and reloads the table. User is able to send only the quantity of items that are in production

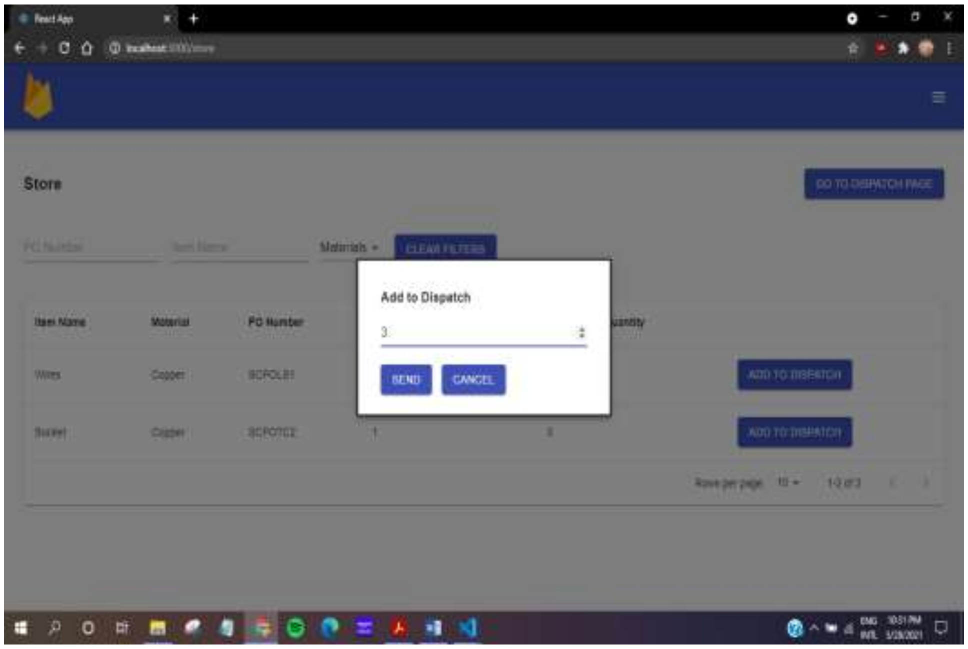
Store Page



Item Name	Material	PO Number	Available Quantity	Dispatched Quantity
Wire	Copper	SCPOL81	3	1

User is able to navigate to this page using the sidebar menu. This page has a table that displays the details of order items that have been produced such as item name, order number, material, available quantity and dispatched quantity.

The table data can be filtered using the fields above the table. As a user enters data in any of fields, data is reloaded in the table according to the results returned from the backend. The “CLEAR FILTERS” button clears the filtering parameters and reloads the table as it was before applying the filters.



Store

GO TO DISPATCH PAGE

Materials CLEAR FILTERS

Add to Dispatch

3

SEND CANCEL

Item Name	Material	PO Number		uantity	
Wires	Copper	SCPOL81			ADD TO DISPATCH
Bucket	Copper	SCPOTC2	1	3	ADD TO DISPATCH

Rows per page: 10 1-2 of 3

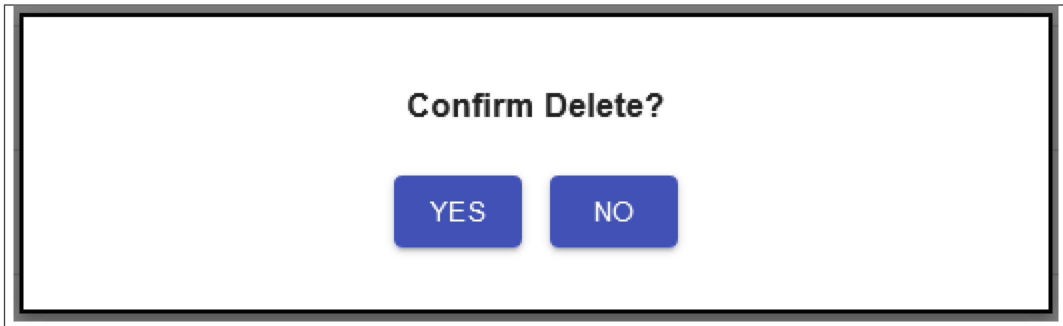
Each row in the table has a “ADD TO DISPATCH” button. When it is clicked a pop-up appears with one field to enter quantity of items to send to production. Clicking on the “SEND” button adds the item to the dispatch list and reloads the table.

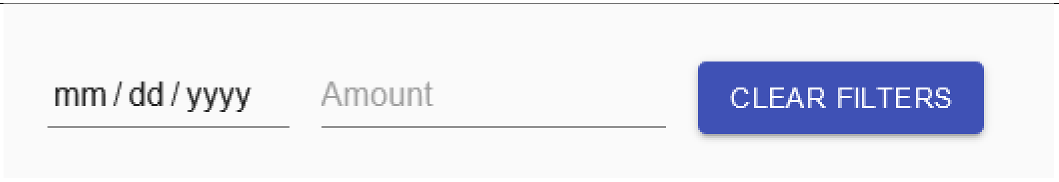
4.2 Operations Manual

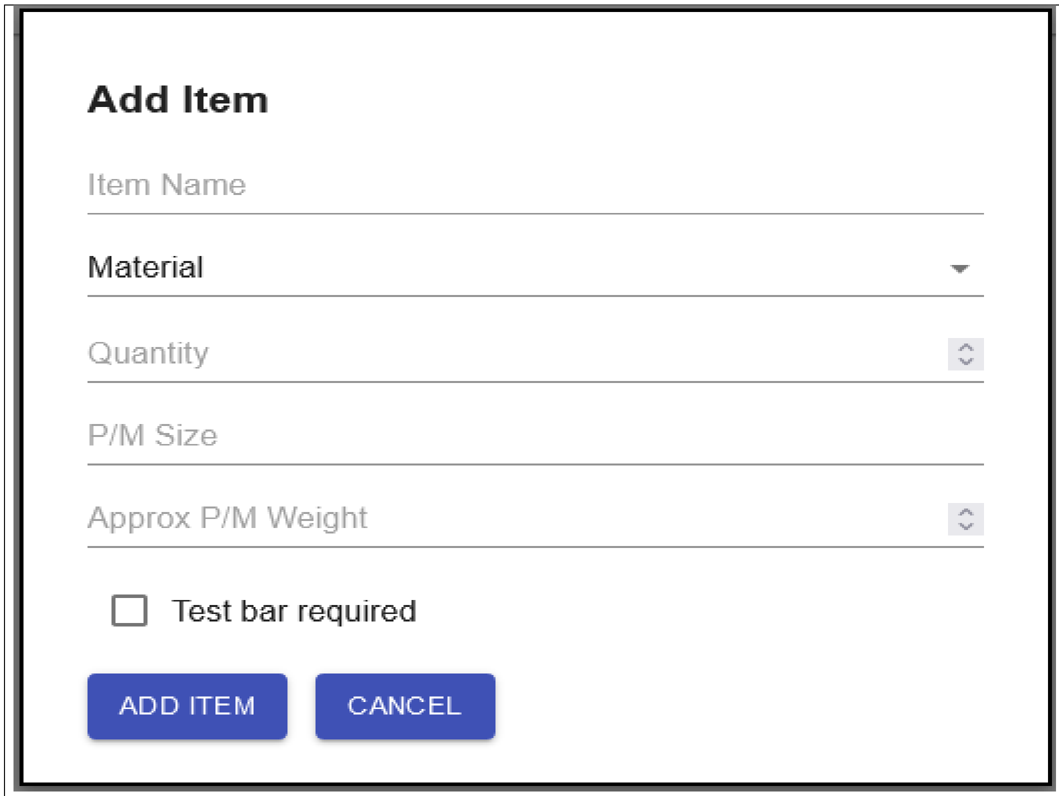
☰	By clicking on this Hamburger icon on the top right area of the sidebar user can access the list of pages in this application and navigate himself

✎	This is the edit icon, user can click on this in selective tables and modify data of that table

🗑	This is the delete icon, user can click on this in tables where has can delete the record/entry if he/she wishes


By clicking on the delete icon user is greeted with a popup displayed above where can click on "YES" to proceed with the delete action or click on "NO" to cancel it


User can utilize filters to filter out data according to his/her requirements, clicking on "CLEAR FILTERS" button resets the filters


Add Item

Item Name

Material

Quantity

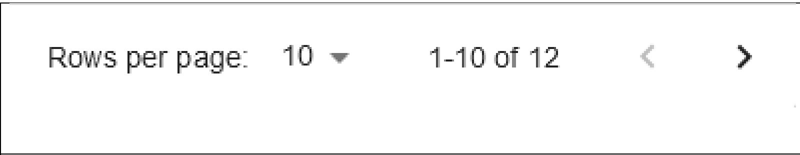
P/M Size

Approx P/M Weight

☐ Test bar required

Inserting of data is done using pop-ups as shown above so that user is not navigated to a new page every time

The pop-ups have 2 buttons at the bottom out of which left button is for submitting data and right one is to Cancel the operation and close the modal


The tables in this application are paginated , user can change the rows per page to view the data according to his needs, by clicking on the arrows user can navigate the pages of the table

4.3 Program Specifications

1)Edit Purchase order

Module	Purchase Order
Program Name	Edit Purchase order
Purpose	Edit data from the order tables.
Input Details	The required fields should not be blank and the user should provide valid data for each field.
Output	The data from the purchase order table is updated.

2)Edit item

Module	Purchase Order
Program Name	Edit item
Purpose	Edit item data from the item table.
Input Details	The required fields should not be blank and the user should provide valid data for each field.
Output	The data from the item table is updated.

3)Add item in purchase order

Module	Purchase Order
Program Name	Add item order
Purpose	Edit data from the order table and add data to item table.
Input Details	The required fields should not be blank and the user should provide valid data for each field.
Output	The data from the purchase order table is updated and data is added to the item table.

4)Delete item from purchase order

Module	Purchase Order
Program Name	Delete item
Purpose	Delete data from the order table and item table.
Input Details	Select item and confirm delete
Output	The data from the item table is updated and data is reloaded.

5)Delete purchase order

Module	Purchase Order
Program Name	Delete order
Purpose	Delete data from the purchase order table.
Input Details	Select the order and confirm delete.
Output	The data from the purchase order table is updated and table is reloaded.

6)Filter purchase order data

Module	Purchase Order
Program Name	Filter purchase order
Purpose	Display data according to some input parameters.
Input Details	Filter fields should be filled.
Output	The data from the purchase order table is displayed according to applied filters.

7)Send to production

Module	Production
Program Name	Send to production
Purpose	Send certain quantity of items to production.
Input Details	Input quantity of items to send to production.
Output	The data from the purchase order table is updated and table is reloaded.

8)Filter items pending production data

Module	Production
Program Name	Filter items pending production
Purpose	Display data according to some input parameters.
Input Details	Filter fields should be filled.
Output	The data from the items pending production is displayed according to applied filters.

9)Send to Store

Module	Production
Program Name	Send to store
Purpose	Send certain quantity of items to store.
Input Details	The required fields should not be blank and the user should provide valid data for each field
Output	The data from the items in production table is updated and table is reloaded.

10)Filter items pending production data

Module	Production
Program Name	Filter items in production
Purpose	Display data according to some input parameters.
Input Details	Filter fields should be filled.
Output	The data from the items in production is displayed according to applied filters.

11)Add to dispatch

Module	Store
Program Name	Add to dispatch
Purpose	Add certain quantity of items to dispatch.
Input Details	The required fields should not be blank and the user should provide valid data for each field
Output	The data from the store table is updated and table is reloaded.

12)Filter store table data

Module	Store
Program Name	Filter store table data
Purpose	Display data according to some input parameters.
Input Details	Filter fields should be filled.
Output	The data from the items in production is displayed according to applied filters.

Drawbacks and Limitations

1)This system is made to be deployed internally and to be used by the company staff because of this the customers cannot place orders online, that process still has to be done by physically visiting the company

2)There is no Authentication or Authorization. Even though this application is going to be used by the company staff, a person with insufficient knowledge or bad intentions can tamper with the data

3)The application does not have internet connectivity. So machine failure may lead to data loss if it is not backed up

Proposed Enhancements

1)Add Authentication and Authorization

2)Add separate customer module so that customer can login from anywhere and place orders

3)Add a backup system which backs up data frequently

4)Application will be updated as per user feedback

Conclusions

The requirements stated by the client have been addressed in this application. The application includes the following

1) Purchase Order Page

A list of purchase orders with fields SrNo, PO Number, Customer Name and Date PO Received is displayed in a table which is paginated with default 10 rows per page. Filters can be applied on the same fields to sort the data using them as parameters. Users are able to change page size to [10, 20, 30, 50, 100]. On clicking the delete button the PO is marked as deleted and not shown in the PO list. On clicking the edit button the user is directed to the Edit PO
page.

2)Edit PO Page:

The user is able to input SrNo, PO Number, Customer Name and Date PO Received. Clicking the “Update” button will update these fields.
On clicking the Add Item button a pop up is displayed to the user with the fields Item name, Material, Quantity, P/M Size, Approx P/M weight, Test bar. On clicking the Add button the item is added to the Item List. The Item List has the same column as Add Item Popup with one extra column where Delete and Edit Icons will be present for each item. On clicking Delete the item is be marked as deleted and removed

from the Item List. On clicking the edit button, the same popup is opened with values prefilled, the button Add will now have the label ‘Update’.

3) Production page:

This page has 2 tabs at the top

1. Pending Production
2. In-Production

Pending Production

This tab displays a list of all the items from all the POs who are
yet to be sent to production. If the pending quantity is 0 the item has been produced/Manufactured and will not be shown here.
The grid will have these columns PO Number, Item name, Material, Ordered Quantity, Produced Quantity, Pending Quantity, Send to Production Button. The grid will be filterable on PO Number, Item Name, Material. On clicking the ‘Send to Production’ the user is asked to input the quantity sent to production.

In-Production

This tab displays a list of all the items from all the POs who are in production. The grid will have these columns PO Number, Item name, Material, Ordered Quantity, Produced Quantity, Pending Quantity, Send to Store Button. The grid will be filterable on PO Number, Item

Name, Material. On clicking the ‘Send to store’ a pop up is displayed with the following fields Cast weight, Furnace Number, Operator Name, Actual P/M Weight, Finish Weight, Status [OK/ REJECTED], Remarks.

4) Stores Page

The stores page will have all the items produced in a grid with the columns Item Name, Material, PO No, Quantity available, Quantity dispatched, Add to dispatch Button. The grid will be filterable on columns Item Name, PO Number and Material. On clicking Add to dispatch the item will be added to the current dispatch list and
the user will be asked to enter the quantity of dispatched items

Bibliography

Websites:

- https://www.nodejs.org/en/
- https://www.mongoosejs.com/docs/guide.html
- https://www.stackoverflow.com/tags/node.js
- https://www.reactjs.org/
- https://www.github.com/nodejs
- https://github.com/reactjs
- https://www.w3schools.com

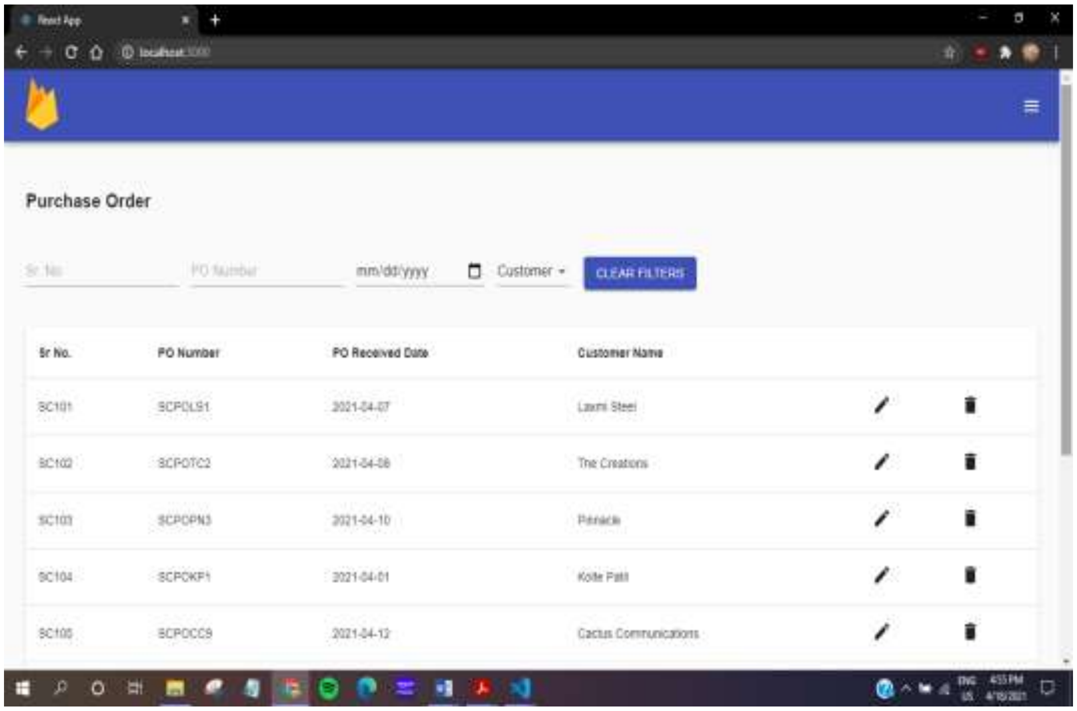
Books:

- Advanced Internet Technologies – Techmax Publications

ANNEXURE 1

USER INTERFACE SCREEN

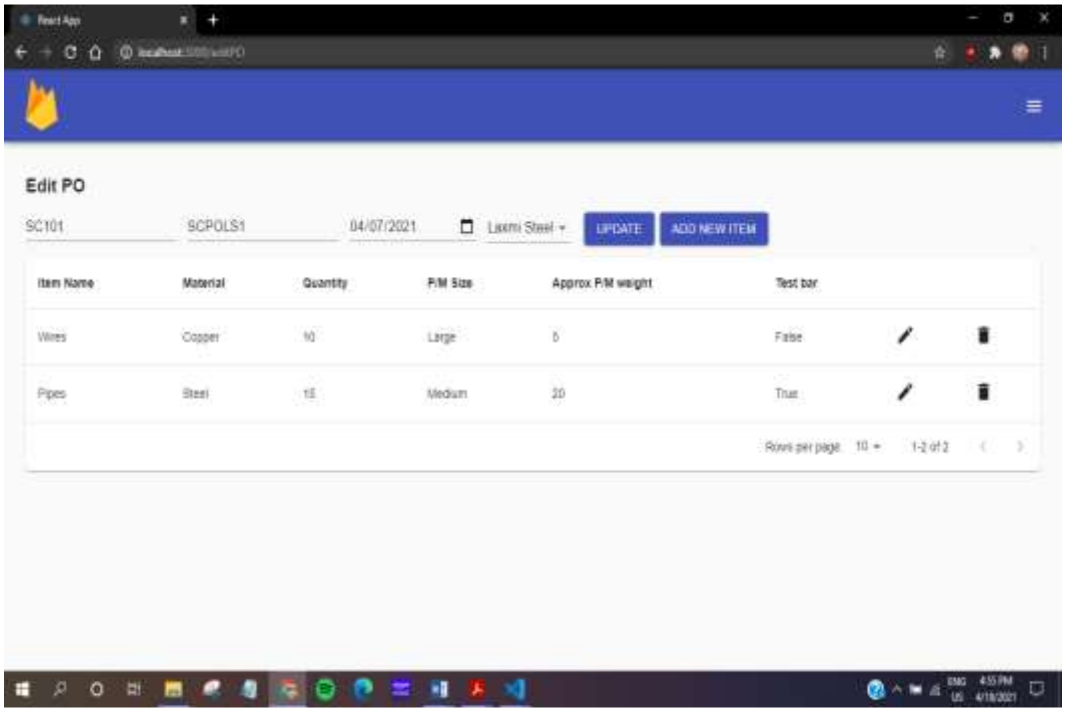
Purchase Order Page



Purchase Order

Sr No.	PO Number	PO Received Date	Customer Name		
SC101	SCPOLS1	2021-04-07	Laxmi Steel		
SC102	SCPOTC2	2021-04-08	The Creations		
SC103	SCPOPN3	2021-04-10	Pinnacle		
SC104	SCPOKP1	2021-04-01	Kolte Patil		
SC105	SCPOCC9	2021-04-12	Cactus Communications		

Edit PO Page



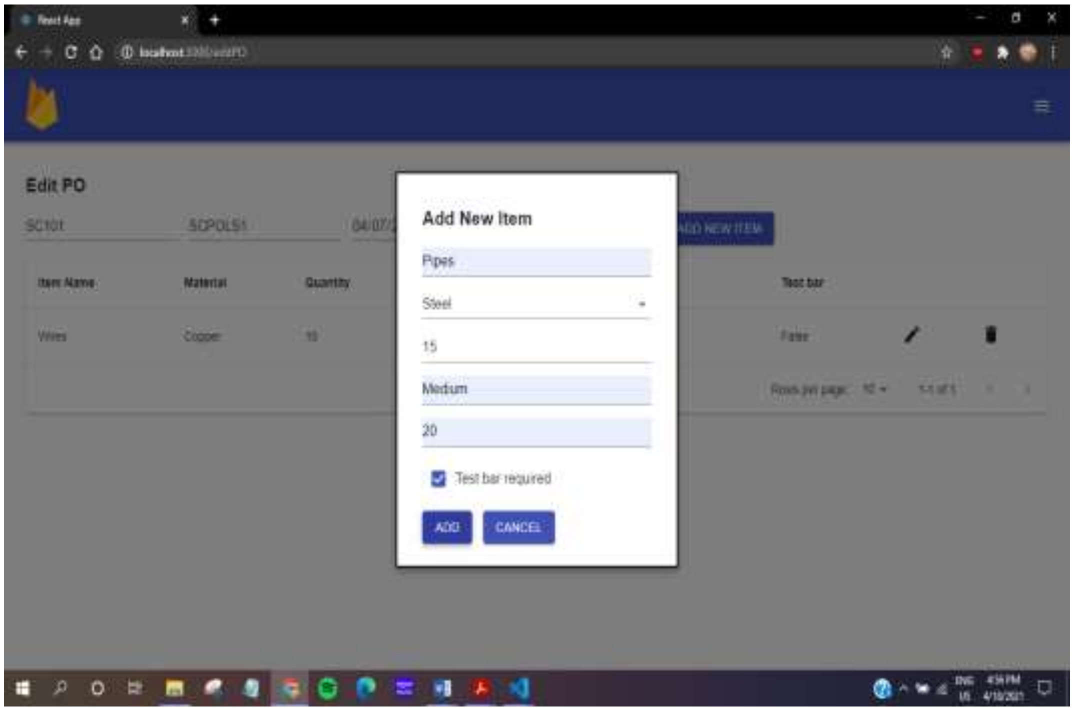
Edit PO

SC101 SCPOLS1 04/07/2021 Laxmi Steel

Item Name	Material	Quantity	P/M Size	Approx P/M weight	Test bar
Wires	Copper	10	Large	5	False
Pipes	Steel	15	Medium	20	True

Rows per page: 10 1-2 of 2

Add new item



Edit PO

SC101 SCPOL51 04/07/

ADD NEW ITEM

Item Name	Material	Quantity	Test bar
Wires	Copper	10	False

Rows per page: 10 1-1 of 1

Add New Item

Pipes

Steel

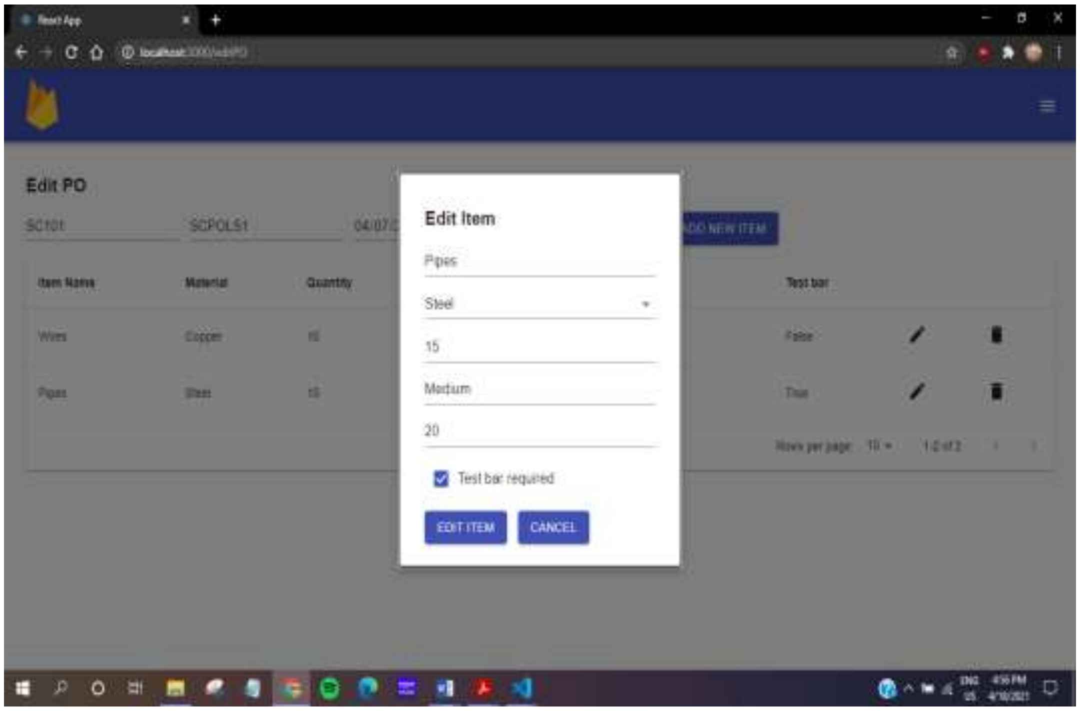
15

Medium

20

[✓] Test bar required

Edit PO Item



Edit PO

SC101 SCPOL51 04/07/

Item Name	Material	Quantity	Test bar
Wires	Copper	15	False
Pipes	Steel	15	True

Rows per page: 10 1-2 of 2

Edit Item

Pipes

Steel

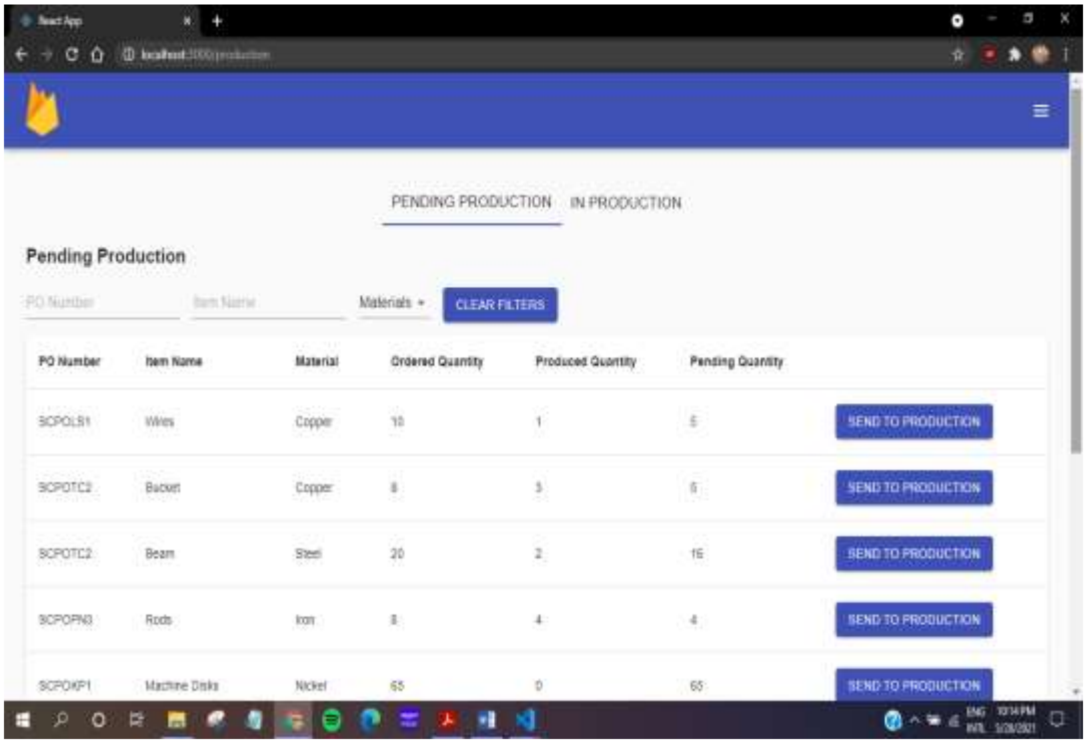
15

Medium

20

[✓] Test bar required

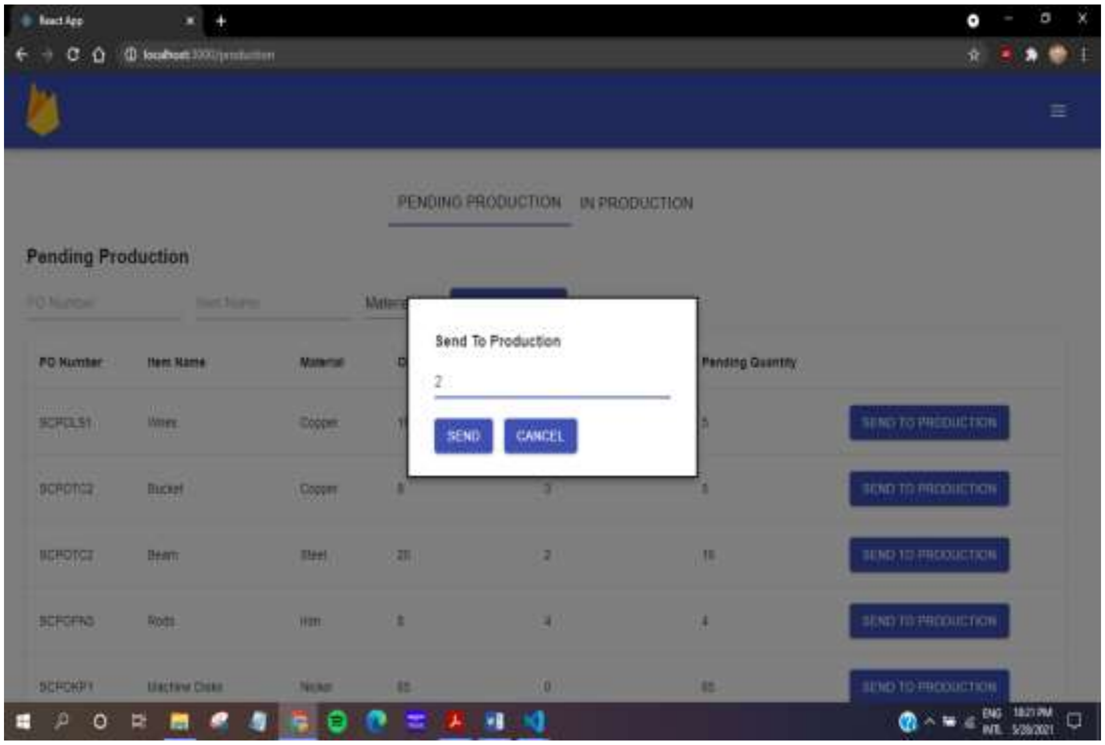
Items pending production



Pending Production

PO Number	Item Name	Material	Ordered Quantity	Produced Quantity	Pending Quantity	
SCPOLS1	Wires	Copper	10	1	5	SEND TO PRODUCTION
SCPOTC2	Bucket	Copper	8	3	5	SEND TO PRODUCTION
SCPOTC2	Beam	Steel	20	2	16	SEND TO PRODUCTION
SCPOPN3	Rods	Iron	8	4	4	SEND TO PRODUCTION
SCPOKP1	Machine Disks	Nickel	65	0	65	SEND TO PRODUCTION

Items being sent to production



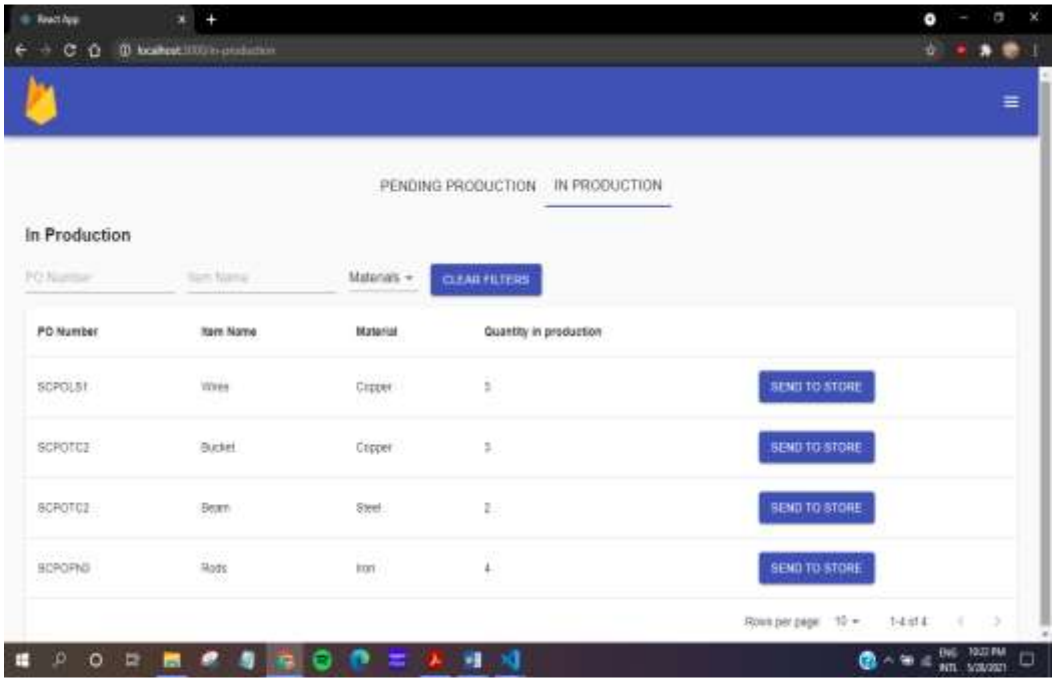
Pending Production

Send To Production

2

PO Number	Item Name	Material	Q		Pending Quantity	
SCPOLS1	Wires	Copper	1		5	SEND TO PRODUCTION
SCPOTC2	Bucket	Copper	8	3	5	SEND TO PRODUCTION
SCPOTC2	Beam	Steel	20	2	18	SEND TO PRODUCTION
SCPOFN3	Rods	Iron	8	4	4	SEND TO PRODUCTION
SCPOKP1	Machine Disks	Nickel	85	0	85	SEND TO PRODUCTION

Items in production

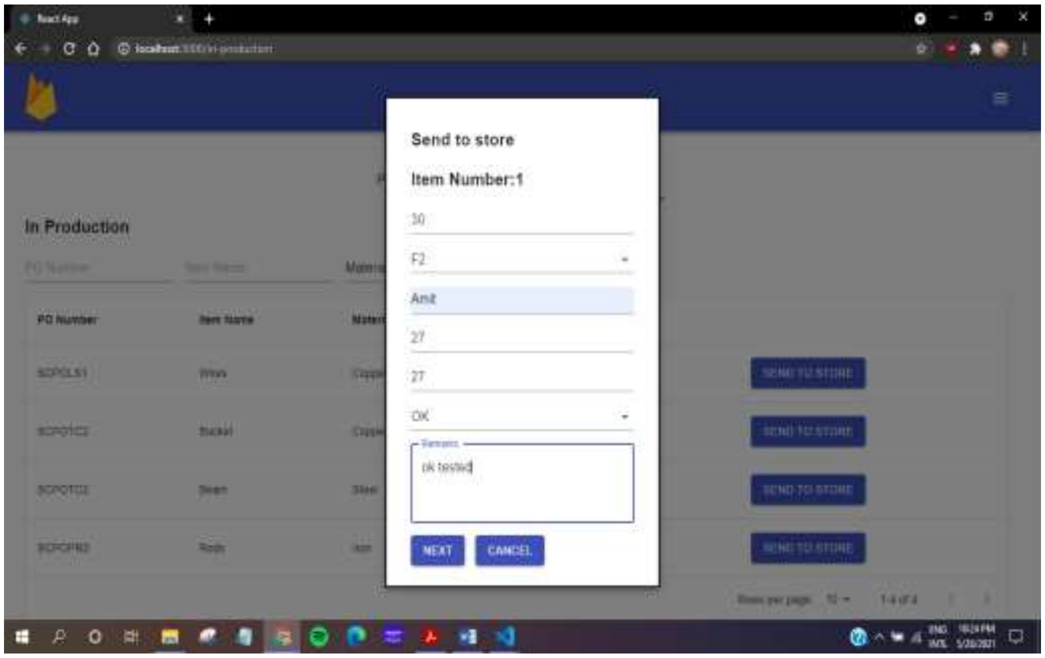


In Production

PO Number	Item Name	Material	Quantity in production	
SCPOLS1	Wires	Copper	3	SEND TO STORE
SCPOTC2	Bucket	Copper	3	SEND TO STORE
SCPOTC2	Beam	Steel	2	SEND TO STORE
SCPOPN3	Rods	Iron	4	SEND TO STORE

Rows per page: 10 1-4 of 4

Add to store



In Production

PO Number	Item Name	Materi
SCPOLS1	Wires	Coppe
SCPOTC2	Bucket	Coppe
SCPOTC2	Beam	Steel
SCPOPN2	Rods	Iron

Send to store

Item Number:1

30

F2

Amit

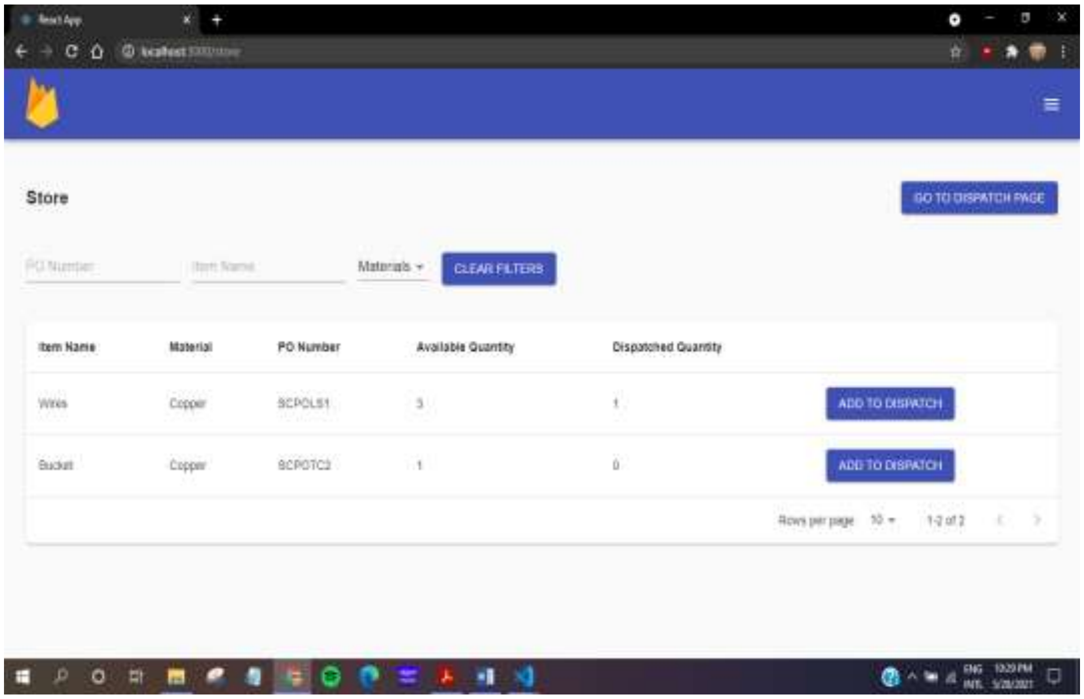
27

27

OK

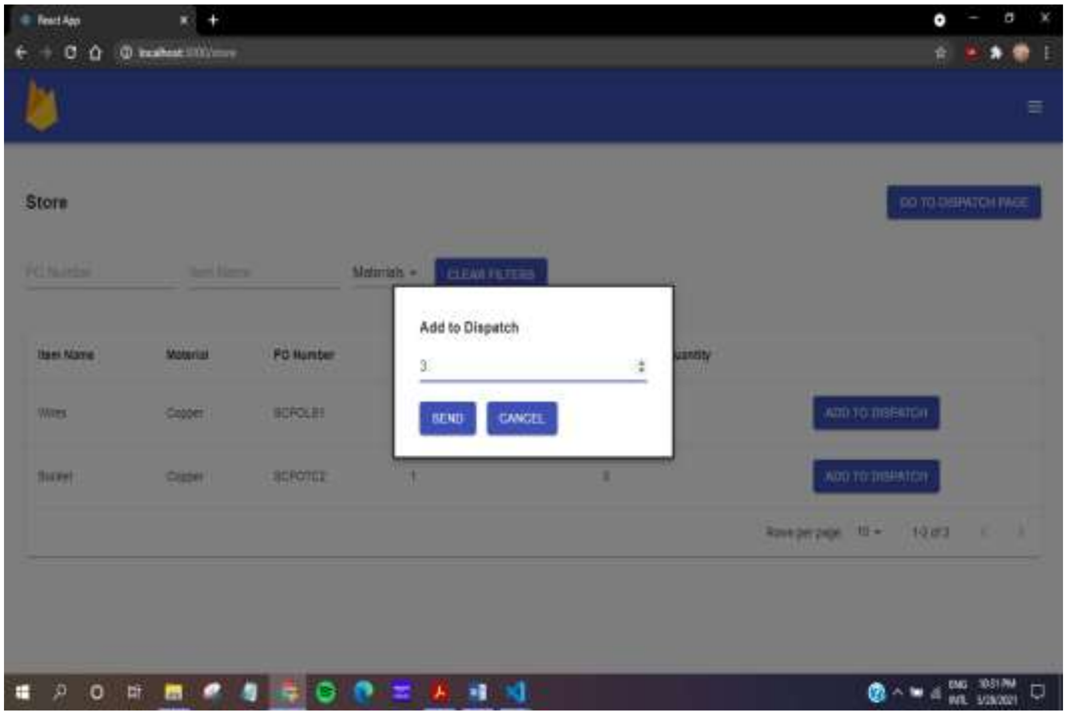
Remarks: ok tested

Store page



Item Name	Material	PO Number	Available Quantity	Dispatched Quantity	
Wires	Copper	SCPOLS1	3	1	ADD TO DISPATCH
Bucket	Copper	SCPOTC2	1	0	ADD TO DISPATCH

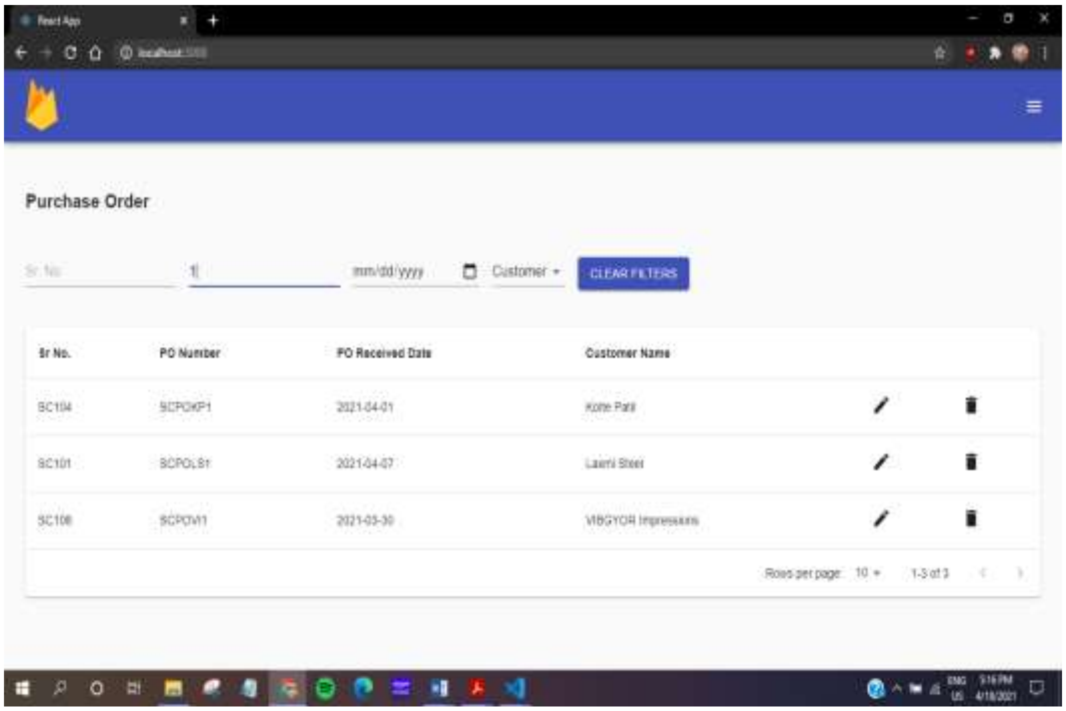
Add to dispatch



ANNEXURE 2

Output Reports with Data

Purchase Order Report

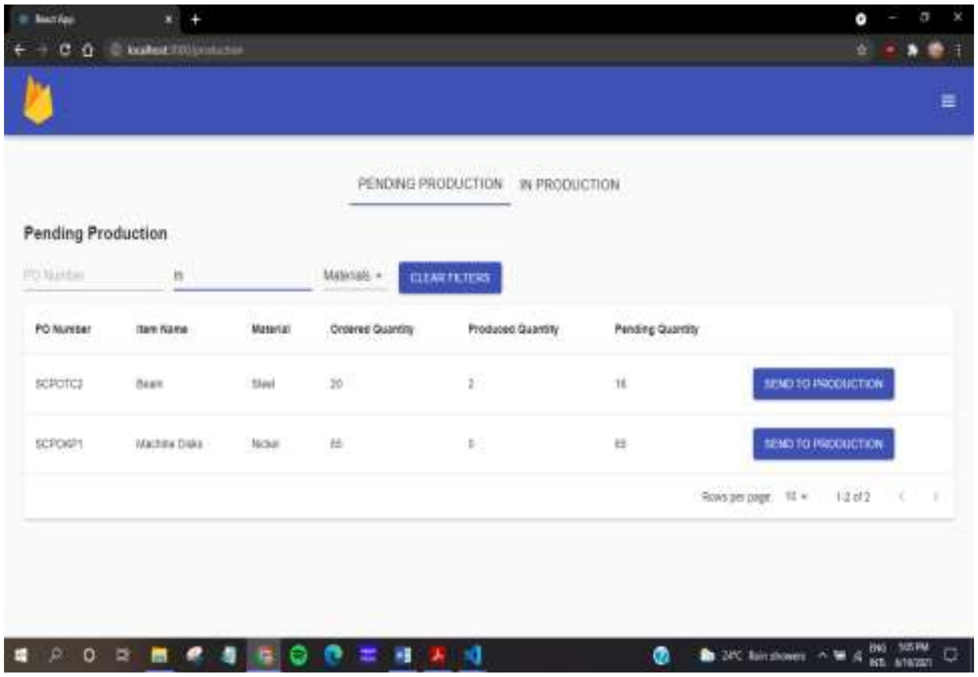


Purchase Order

Sr No.	PO Number	PO Received Date	Customer Name
SC104	SCPOKP1	2021-04-01	Kohe Patil
SC101	SCPOLS1	2021-04-07	Laxmi Steel
SC106	SCPOVI1	2021-03-30	VIBGYOR Impressions

Rows per page: 10 1-3 of 3

Items Pending Production Report



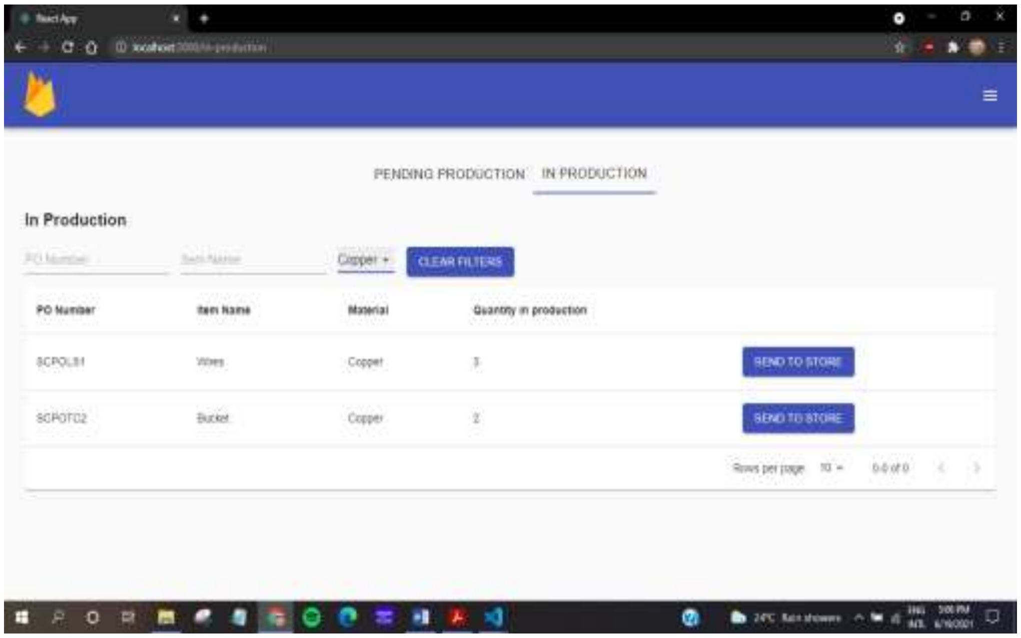
Pending Production

PO Number | ts | Materials | CLEAR FILTERS

PO Number	Item Name	Material	Ordered Quantity	Produced Quantity	Pending Quantity	
SCPOTC2	Beam	Steel	20	2	18	SEND TO PRODUCTION
SCPOKP1	Machine Disks	Nickel	65	0	65	SEND TO PRODUCTION

Rows per page: 10 1-2 of 2

Items in production report

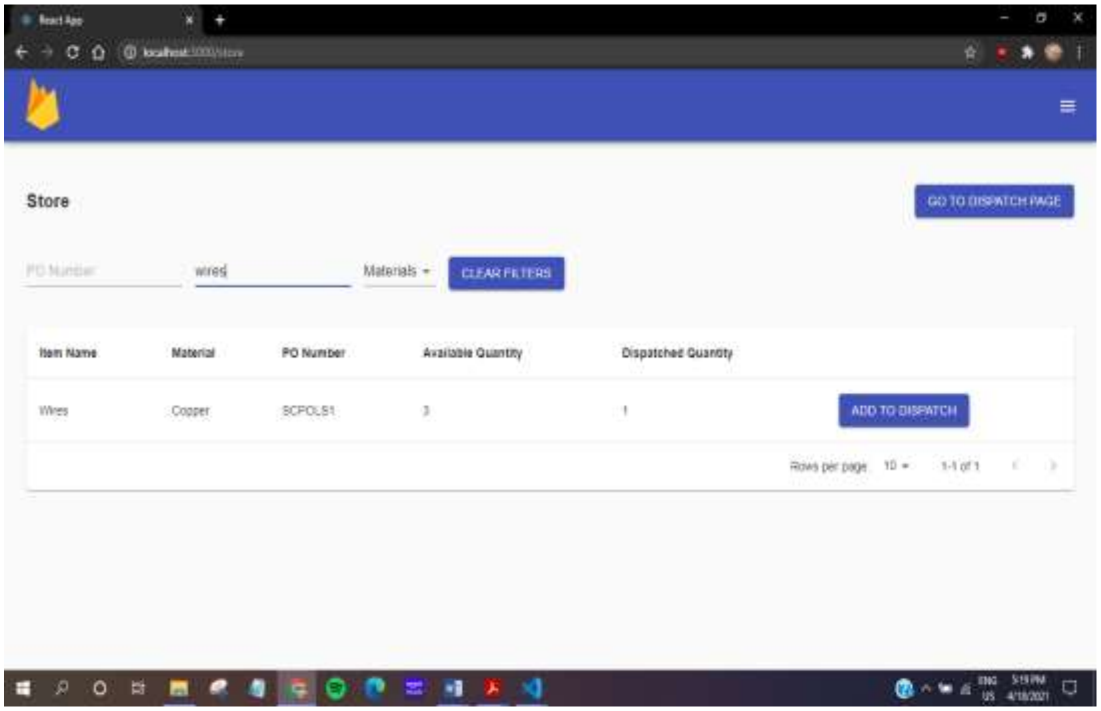


In Production

PO Number	Item Name	Material	Quantity in production
SCPOLS1	Wires	Copper	3
SCPOTC2	Bucket	Copper	2

Rows per page 10 0-0 of 0

Stores Report



Item Name	Material	PO Number	Available Quantity	Dispatched Quantity	
Wires	Copper	SCPOLS1	3	1	ADD TO DISPATCH

ANNEXURE 3

SAMPLE PROGRAM CODE

1)edit-item-modal.tsx

```
import Modal from "../../../components/modal-component/modal-component";

import { useForm, Controller } from "react-hook-form";

import { useState } from "react";

import { Button, Select, MenuItem } from "@material-ui/core";

import TextField from "@material-ui/core/TextField";
import FormControlLabel from "@material-
ui/core/FormControlLabel";

import Checkbox from "@material-ui/core/Checkbox";

import { useDispatch } from "react-redux";

import { editItemAction } from "../purchase-order.action";

import AlertDialog from "../../../components/alert-component/alert-
component";


const EditItemModal = (props: any) => {

  const dispatch = useDispatch();

  const [itemEditedAlertState, setItemEditedAlertState] =
useState(false);

  const closeAlert = () => {

    setItemEditedAlertState(false);

  };


  const modalSubmit = async (data: any) => {

    props.passedRowData.itemsList[props.rowIndex] = {

      ...props.passedRowData.itemsList[props.rowIndex],
      itemName: data.itemName,
      quantity: data.quantity,
      approxPmWeight: data.approxPmWeight,
      pmSize: data.pmSize,
    };

    const selectedMaterial = data.material.split("_");

    props.passedRowData.itemsList[props.rowIndex].material = {
      ...props.passedRowData.itemsList[props.rowIndex].material,
      _id: selectedMaterial[0],
      materialName: selectedMaterial[1],
    };

    if (props.itemData._id) {
      data._id = props.itemData._id;
      dispatch(editItemAction({  ...data,  material:  selectedMaterial[0]
},props.passedRowData._id));
```

```
    }
    props.modalHandler();
    setItemEditedAlertState(true);
  };

  const { register, errors, control, handleSubmit } = useForm();
  return (
    <div>
      <Modal                    modalState={props.modalState}
modalHandler={props.modalHandler}>
        <h2>Edit Item</h2>
        <form
          key={2}
          onSubmit={handleSubmit(modalSubmit)}
          className="editItem-form"
        >
          <TextField
            className="editItemFields"
            type="text"
            placeholder="Item Name"
            name="itemName"
            defaultValue={props.itemData.itemName}
            inputRef={register({
              required: "Item Name is required",
            })}
          />
          {errors.itemName && <p>{errors.itemName.message}</p>}
          <Controller
            as={
              <Select className="editItemFields" required>
                <MenuItem value="" disabled>
                  Material
                </MenuItem>
                {props.dropDownData.map((material: any) => (
                  <MenuItem
                    key={material._id}
```

```
              value={`${material._id}_${material.materialName}`}
            >
              {material.materialName}
            </MenuItem>
          ))}
        </Select>
      }
      name="material"
      rules={{ required: "Material Name is required" }}
      control={control}

defaultValue={`${props.itemData.material?._id}_${props.itemData.
material?.materialName}`}
     />


     <TextField
      className="editItemFields"
      type="number"
```

```
  placeholder="Quantity"
  name="quantity"
  defaultValue={props.itemData.quantity}
  inputRef={register({
    required: "Quantity is required",
  })}
/>
{errors.quantity && <p>{errors.quantity.message}</p>}
<TextField
  className="editItemFields"
  type="text"
  placeholder="P/M Size"
  name="pmSize"
  defaultValue={props.itemData.pmSize}
  inputRef={register({
    required: "P/M Size is required",
  })}
/>
```

```
{errors.pmSize && <p>{errors.pmSize.message}</p>}
<TextField
  className="editItemFields"
  type="text"
  placeholder="Approx P/M Weight"
  name="approxPmWeight"
  defaultValue={props.itemData.approxPmWeight}
  inputRef={register({
    required: "Approx P/M Weight is required",
  })}
/>
{errors.approxPmWeight                                  &&
<p>{errors.approxPmWeight.message}</p>}
<FormControlLabel
  className="editItemFields"
  control={
    <Checkbox
      defaultChecked={props.itemData.testBar}
      color="primary"
      name="testBar"
      inputRef={register({})}
    />
  }
  label="Test bar required"
/>
<section>
  <Button
    type="submit"
    variant="contained"
    color="primary"
    className="editItemButton"
  >
    Edit Item
  </Button>
  <Button
    className="editItemButton"
```

```
            variant="contained"
            color="primary"
            onClick={props.modalHandler}
           >
            Cancel
           </Button>
          </section>
         </form>
        </Modal>
        <AlertDialog
         alertState={itemEditedAlertState}
         alertClose={closeAlert}
         title="Item Edited"
         content="Item edited successfully in the Item List"
        />
       </div>
      );
     };
```

```
export default EditItemModal;
```

2)purchase-order.service.ts

```
import {
  IPurchaseOrderDetails,
  IUpdatePurchaseOrderDetails,
} from "./purchase-order.interface";
import { IFilter, Pagination } from "../utility/app-interfaces";
import { ERROR_CODES, GlobalErrors } from "../errors/errors";
import {
  createPurchaseOrder,
  getPurchaseOrder,
  deletePurchaseOrder,
  updatePurchaseOrder,
} from "./purchase-order.repository";
import { ObjectID } from "../utility/utility";
```

```
export const createPurchaseOrderService = async (
  purchaseOrderDetails: IPurchaseOrderDetails
) => {
  purchaseOrderDetails.itemsList.forEach((item: any) => {
    item.pendingQuantity = item.quantity;
  });
  const result: any = await
createPurchaseOrder(purchaseOrderDetails);
  if (!result) {
    throw ERROR_CODES[GlobalErrors.FAILED_TO_CREATE];
  }
  return result;
};

export const getPurchaseOrderService = async (
  filters: any,
  paginationDetails: any
```

```
) => {
  const skip = paginationDetails.pages * paginationDetails.rowsPerPage;
  const limit = paginationDetails.rowsPerPage;
  const Filters = setFilters(filters);
  const result = await getPurchaseOrder(Filters, { skip, limit });
  if (result.purchaseOrder.length === 0) {
    throw ERROR_CODES[GlobalErrors.NO_RECORD_FOUND];
  }

  const response = result.purchaseOrder.map((record: any) => {
    return {
      ...record.toObject(),
      poReceivedDate: new Date(record.poReceivedDate)
        .toISOString()
        .slice(0, 10),
    };
  });
  return {response,recordCount:result.recordCount};
};

export const deletePurchaseOrderService = async (poId: string) => {
  const result = await deletePurchaseOrder(ObjectID(poId));
  if (result.n === 0) {
    throw ERROR_CODES[GlobalErrors.FAILED_TO_DELETE];
  }
  return result;
};

export const udatePurchaseOrderService = async (
  purchaseOrderDataToUpdate: IUpdatePurchaseOrderDetails,
  poId: string
) => {
  const result = await
updatePurchaseOrder(purchaseOrderDataToUpdate, poId);
  if (result.n === 0) {
    throw ERROR_CODES[GlobalErrors.FAILED_TO_UPDATE];
  }
  return result;
};
```