

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

REPORT ON MINI PROJECT

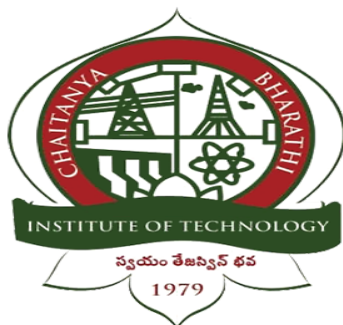
Project Name:

WEB APPLICATION FOR INVENTORY
MANAGEMENT

Student & Mentor Details

S.No	Student Name	Roll.No	Mentor Name
1.	M. Akshay Kumar	160118733026	Smt. Isha padhy
2.	M. Subramani	160118733051	Sri K. Kiran prakash
3.	Venkat sai teja	160118733055	Smt. Isha padhy

**Department of Computer Science and Engineering,
Chaitanya Bharathi Institute of Technology (Autonomous),
(Affiliated to Osmania University, Hyderabad)
Hyderabad, TELANGANA (INDIA) – 500 075
November-2020**



CERTIFICATE

This is to certify that the project titled “**Web Application for Inventory Management**” is the bonafide work carried out by **Akshay Kumar Malathkar(160118733026), Subramani Murugesan(160118733051), Venkat Sai Teja(160118733055)** students of B.E.(CSE) of Chaitanya Bharathi Institute of Technology, Hyderabad, affiliated to Osmania University, Hyderabad, Telangana(India) during the academic year 2020-2021, submitted in partial fulfillment of the requirements for the award of the degree in **Bachelor of Engineering (Computer Science and Engineering)** and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

Supervisors

Smt. Isha Padhy, Asst.Professor
Sri K. Kiran Prakash, Asst.Professor

Head, CSE Dept.

Dr Y Ramadevi
Dept of CSE,CBIT

Place: Hyderabad

Date: 29-11-2020

External Examiner

DECLARATION

I/we hereby declare that the project entitled “**Web Application for Inventory Management**” submitted for the B. E (CSE) degree is our original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

Name(s) and Signature(s) of the Student

Akshay Kumar Malathkar(160118733026)

Subramani Murugesan(160118733051)

Venkat Sai Teja(160118733055)

Place: Hyderabad

Date: 29-11-2020

Table of Contents

ABSTRACT.....	4
INTRODUCTION.....	5
SCOPE OF THE PROJECT.....	5
SOFTWARE SPECIFICATIONS.....	6
FRONT-END DESIGN	6
BACK-END	7
DATABASE.....	8
LARAVEL	16
FLOW-CHART.....	23
RESULTS AND DISCUSSIONS	24
INVENTORY	26
INVOICE GENERATION	28
SALES ANALYSIS	31
INSIGHTS.....	33
FUTURE WORK	35
CONCLUSION.....	36
REFERENCES	36

ABSTRACT

As a business grows, the number of transactions increases and keeping track of all the transactions and the products involved will be difficult and may lead to discrepancies. This web application helps both the manufacturers and the sellers to keep track of their transactions and products. This application also creates a platform for the manufacturers to sell their products directly to the sellers.

The product information is stored in a database and the day to day sales are monitored. Given a date range, a user can view the transactions made in that particular period of time. Sales analysis is done to help the seller know if they are making a profit or not.

When a seller is running low on a specific product, they can request the manufacturer for that product. Invoices can be generated by the sellers at the time of transactions.

Finally, this web application makes inventorying easy and provides additional information regarding the sales. By using this information, a seller can maximize their profits. And all these features can be accessed from anywhere and at any time.

INTRODUCTION

After analyzing many existing IMS we have now the obvious vision of the project to be developed. Before we started to build the application team had many challenges. We defined our problem statement as:

- To make a web application of IMS for a small organization.
- To make the system easily managed and can be secured.
- To cover all the areas of IMS like purchase details, sales details and stock management and invoice generation.

SCOPE OF THE PROJECT:

Inventory Management System (IMS) is targeted to the small or medium organization which doesn't have many godown or warehouses i.e. only to those organizations that have a single power of authority. Some of the scopes are: Only one person is responsible in assigning the details or records It is security driven. Godown can be added as per the requirement.

SOFTWARE SPECIFICATIONS

FRONT-END DESIGN:

HTML, CSS, Bootstrap, JavaScript, jQuery

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification.

Presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

JavaScript, often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

BOOTSTRAP: Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

jQuery: jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

BACK-END:

Database: MYSQL

The below diagram shows the tables in our database “inventory_management”.

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> customer_details	★ Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> items_sold_manufacturer	★ Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> items_sold_seller	★ Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> item_requests	★ Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> manufacturer_inventory	★ Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> migrations	★ Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> seller_inventory	★ Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> signup_manufacturers	★ Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> signup_sellers	★ Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
9 tables	Sum	36	InnoDB	utf8mb4_general_ci	144.0 KiB	0 B

Migrations in Laravel:

Migrations are like version control for your database, allowing your team to modify and share the application's database schema. Migrations are typically paired with Laravel's schema builder to build your application's database schema.

If you have ever had to tell a teammate to manually add a column to their local database schema, you've faced the problem that database migrations solve.

The Laravel Schema facade provides database agnostic support for creating and manipulating tables across all of Laravel's supported database systems.

Creating Columns:

The table method on the Schema facade may be used to update existing tables. Like the create method, the table method accepts two arguments: the name of the table and a Closure that receives a Blueprint instance you may use to add columns to the table:

```
Schema::table('users', function (Blueprint $table) {  
    $table->string('email');  
});
```

Running Migrations:

To run all of your outstanding migrations, execute the migrate Artisan command:

```
php artisan migrate
```

signup_manufacturers:

This table stores the details of the manufacturers who have registered on our application.

The following code creates a table with the columns specified.

```
public function up()  
{  
    Schema::create('signup_manufacturers', function (Blueprint $table) {
```

```

    $table->increments('id');

    $table->string('full_name');

    $table->string('email');

    $table->string('password');

    $table->timestamps();

});

}

```

		id	full_name	email	password	created_at	updated_at
<input type="checkbox"/>	Edit	1	Sample User	sampleuser@gmail.com	eyJpdil6InpFelFuc25GQzcrTXJKNE9BLzBVOUE9PSIslnZhbH...	2020-10-31 15:41:52	2020-10-31 15:41:52
<input type="checkbox"/>	Edit	6	sample user1	sample1user@gmail.com	eyJpdil6jVjaitwY2QwbEltRkJlUUcxSnVwUUE9PSIslnZhbH...	2020-11-08 10:31:38	2020-11-08 10:31:38

signup_sellers:

This table stores the details of the sellers who have registered on our application.

The following code creates a table with the columns specified.

```

public function up()
{
    Schema::create('signup_sellers', function (Blueprint $table) {
        $table->increments('id');

        $table->string('full_name');

        $table->string('email');

        $table->string('password');

        $table->timestamps();

    });
}

```

				id	full_name	email	password	created_at	updated_at
<input type="checkbox"/>	Edit	Copy	Delete	4	Seller-1	seller1@gmail.com	eyJpdil6lkhqR0gzcVVsV1JVWEY3UmxBScmpOZVE9PSIsInZhbH...	2020-11-15 16:21:09	2020-11-15 16:21:09
<input type="checkbox"/>	Edit	Copy	Delete	5	Seller-2	seller2@gmail.com	eyJpdil6lIR3RkNLeMhSS3BiUkMxUnM3dEdPZ1E9PSIsInZhbH...	2020-11-15 16:58:45	2020-11-15 16:58:45

manufacturer_inventory:

This table stores the details of the products stored by the manufacturer in their inventory.

The following code creates a table with the columns specified.

```
public function up()
{
    Schema::create('manufacturer_inventory', function (Blueprint $table) {
        $table->increments('id');
        $table->string('product_name');
        $table->integer('quantity');
        $table->integer('production_cost');
        $table->integer('material_cost');
        $table->integer('price');
        $table->date('date_of_manufacture');
        $table->date('expiry_date');
        $table->string('name_of_manufacturer');
        $table->string('email');
        $table->timestamps();
    });
}
```

id	product_name	quantity	production_cost	material_cost	price	date_of_manufacture	expiry_date	name_of_manufacturer	email
7	sample_product	4	100	200	1200	2020-11-08	2020-12-05	sample user1	sample1user@gmail.com
8	Product-2	1	1200	700	2400	2020-11-15	2020-12-12	Sample User	sampleuser@gmail.com
9	Product-1	10	1000	200	1500	2020-11-23	2020-12-12	Sample User	sampleuser@gmail.com

seller_inventory:

This table stores the details of the products stored by the seller in their inventory.

The following code creates a table with the columns specified.

```
public function up()
{
    Schema::create('seller_inventory', function (Blueprint $table) {
        $table->increments('id');
        $table->string('product_name');
        $table->integer('quantity');
        $table->integer('price');
        $table->integer('selling_price');
        $table->date('date_of_manufacture');
        $table->date('expiry_date');
        $table->string('name_of_manufacturer');
        $table->string('email');
        $table->timestamps();
    });
}
```

id	product_name	quantity	price	selling_price	date_of_manufacture	expiry_date	name_of_manufacturer	email
6	Product-3	6	1900	5000	2020-11-05	2020-11-28	Akshay Kumar	aakashkumar1080@gmail.com
17	9 quantity	3	3000	-1	2020-11-07	2020-12-05	Sample User	aakashkumar1080@gmail.com
19	sample_product	6	1200	2890	2020-11-08	2020-12-05	sample user1	akshaykumar200042@gmail.com
23	Product-2	2	2400	-1	2020-11-15	2020-12-12	Sample User	seller2@gmail.com
25	Product-5	4	1000	2100	2020-11-18	2021-01-23	Manufacturer-4	seller1@gmail.com
26	Product-2	1	2400	3200	2020-11-15	2020-12-12	Sample User	seller1@gmail.com

item_requests:

This table stores the details of the products requested by the seller to a particular manufacturer.

The following code creates a table with the columns specified.

public function up()

```
{
    Schema::create('item_requests', function (Blueprint $table) {
        $table->increments('id');
        $table->integer('manufacturer_product_id');
        $table->string('sellers_name');
        $table->string('sellers_email');
        $table->string('manufacturers_email');
        $table->string('product_name');
        $table->integer('quantity_required')->unsigned()->nullable();
        $table->timestamps();
    });
}
```

id	manufacturer_product_id	sellers_name	sellers_email	manufacturers_email	product_name	quantity_required
21	7	Seller-2	seller2@gmail.com	sample1user@gmail.com	sample_product	2
23	7	Seller-1	seller1@gmail.com	sample1user@gmail.com	sample_product	3
24	8	Seller-1	seller1@gmail.com	sampleuser@gmail.com	Product-2	1

items_sold_manufacturer:

This table stores the details of the products sold by the manufacturer to the sellers.

The following code creates a table with the columns specified.

```
public function up()
{
    Schema::create('items_sold_manufacturer', function (Blueprint $table) {
        $table->increments('id');
        $table->string('product_name');
        $table->integer('quantity');
        $table->integer('price');
        $table->integer('selling_price');
        $table->date('date_of_manufacture');
        $table->date('expiry_date');
        $table->string('name_of_manufacturer');
        $table->string('sellers_email');
        $table->string('manufacturers_email');
        $table->date('date_of_transaction')->nullable();
        $table->timestamps();
    });
}
```

```
});

}
```

id	product_name	quantity	price	selling_price	date_of_manufacture	expiry_date	name_of_manufacturer	sellers_email	manufacturers_email	date_of_transaction
5	Product-2	5	1900	2400	2020-11-15	2020-12-12	Sample User	seller1@gmail.com	sampleuser@gmail.com	2020-11-14
6	Product-1	1	1900	2000	2020-11-06	2020-12-07	Sample User	seller1@gmail.com	sampleuser@gmail.com	2020-11-15
7	Product-2	2	1900	2400	2020-11-15	2020-12-12	Sample User	seller2@gmail.com	sampleuser@gmail.com	2020-11-15

items_sold_seller:

This table stores the details of the products sold by the seller to the customers.

The following code creates a table with the columns specified.

```
public function up()
{
    Schema::create('items_sold_seller', function (Blueprint $table) {
        $table->increments('id');
        $table->integer('seller_product_id');
        $table->string('product_name');
        $table->integer('quantity_sold')->unsigned()->nullable();
        $table->integer('price');
        $table->integer('selling_price');
        $table->date('date_of_manufacture');
        $table->date('expiry_date');
        $table->string('name_of_manufacturer');
        $table->string('sellers_email');
```

```

    $table->string('customer_mobile');

    $table->integer('sold');

    $table->timestamps();

});

}

```

id	seller_product_id	product_name	quantity_sold	price	selling_price	date_of_manufacture	expiry_date	name_of_manufacturer	sellers_email	customer_mobile	sold
12	16	9 quantity	4	3000	2800	2020-11-07	2020-12-05	Sample User	akshaykumar200042@gmail.com	9347644865	1
14	18	5 quantity	2	2020	2500	2020-11-06	2020-11-07	Sample User	akshaykumar200042@gmail.com	7013358682	1
16	22	Product-1	1	2000	2100	2020-11-06	2020-12-07	Sample User	seller1@gmail.com	9728347823	1
26	25	Product-5	5	1000	2100	2020-11-18	2021-01-23	Manufacturer-4	seller1@gmail.com	9728347823	1
27	26	Product-2	1	2400	3200	2020-11-15	2020-12-12	Sample User	seller1@gmail.com	9728347823	1

customer_details:

This table stores the details of the customers added by the sellers to whom the products are going to be sold.

The following code creates a table with the columns specified.

```

public function up()
{
    Schema::create('customer_details', function (Blueprint $table) {
        $table->increments('id');

        $table->string('customer_name');

        $table->string('customer_mobile');

        $table->date('date_of_transaction');

        $table->string('sellers_email');

        $table->integer('sold');

        $table->timestamps();

    });
}

```


}

id	customer_name	customer_mobile	date_of_transaction	sellers_email	sold
1	Peter	7013358682	2020-11-05	akshaykumar200042@gmail.com	1
2	John	8019270918	2020-11-06	akshaykumar200042@gmail.com	1
10	Customer-1	9728347823	2020-11-16	seller1@gmail.com	1
18	Customer-1	9728347823	2020-11-23	seller1@gmail.com	1

LARAVEL:

Laravel is a free, open-source PHP web framework, created by Taylor Otwell and intended for the development of web applications following the model–view–controller (MVC) architectural pattern and based on Symphony. Some of the features of Laravel are a modular packaging system with a dedicated dependency manager, different ways for accessing relational databases, utilities that aid in application deployment and maintenance, and its orientation toward syntactic sugar.

The source code of Laravel is hosted on GitHub and licensed under the terms of MIT License.

LARAVEL-INSTALLATION:

Server Requirements:

The Laravel framework has a few system requirements. All of these requirements are satisfied by the [Laravel Homestead](#) virtual machine, so it's highly recommended that you use Homestead as your local Laravel development environment.

However, if you are not using Homestead, you will need to make sure your server meets the following requirements:

- PHP >= 7.3

- BCMath PHP Extension
- Ctype PHP Extension
- Fileinfo PHP Extension
- JSON PHP Extension
- Mbstring PHP Extension
- OpenSSL PHP Extension
- PDO PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension

INSTALLATION PROCESS:

Via Laravel Installer

First, download the Laravel installer using Composer:

```
--composer global require laravel/installer
```

Make sure to place Composer's system-wide vendor bin directory in your \$PATH so the laravel executable can be located by your system. This directory exists in different locations based on your operating system

ROUTING:

The most basic Laravel routes accept a URI and a **Closure**, providing a very simple and expressive method of defining routes:

```
Route::get('foo', function () {  
    return 'Hello World';  
});
```

Available Router Methods:

The router allows you to register routes that respond to any HTTP verb:

```
Route::get($uri, $callback);  
Route::post($uri, $callback);  
Route::put($uri, $callback);  
Route::patch($uri, $callback);  
Route::delete($uri, $callback);  
Route::options($uri, $callback);
```

SESSIONS IN LARAVEL:

Since HTTP driven applications are stateless, sessions provide a way to store information about the user across multiple requests. Laravel ships with a variety of session backends that are accessed through an expressive, unified API. Support for popular backends such as Memcached, Redis, and databases is included out of the box.

Storing Data:

To store data in the session, you will typically use the put method or the session helper:

```
// Via a request instance...  
$request->session()->put('key', 'value');  
  
// Via the global helper...  
session(['key' => 'value']);
```

Retrieving Data:

There are two primary ways of working with session data in Laravel: the global session helper and via a Request instance. First, let's look at accessing the session via a Request instance, which can be type-hinted on a controller method.

Remember, controller method dependencies are automatically injected via the Laravel service container:

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use App\Http\Controllers\Controller;
```

```
use Illuminate\Http\Request;
```

```
class UserController extends Controller
```

```
{
```

```
    /**
```

```
     * Show the profile for the given user.
```

```
     *
```

```
     * @param Request $request
```

```
     * @param int $id
```

```
     * @return Response
```

```
    */
```

```
    public function show(Request $request, $id)
```

```

    {
        $value = $request->session()->get('key');

        //
    }
}

```

CONTROLLERS:

Below is an example of a basic controller class. Note that the controller extends the base controller class included with Laravel. The base class provides a few convenience methods such as the middleware method, which may be used to attach middleware to controller actions:

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use App\Http\Controllers\Controller;
```

```
use App\Models\User;
```

```
class UserController extends Controller
```

```
{
```

```
    /**
```

```
     * Show the profile for the given user.
```

```
     *
```

```
     * @param int $id
```

```

    * @return \Illuminate\View\View
    */
    public function show($id)
    {
        return view('user.profile', ['user' =>
User::findOrFail($id)]);
    }
}

```

You can define a route to this controller action like so:

```
use App\Http\Controllers\UserController;
```

```
Route::get('user/{id}', [UserController::class, 'show']);
```

Now, when a request matches the specified route URI, the **show** method on the **UserController** class will be executed. The route parameters will also be passed to the method.

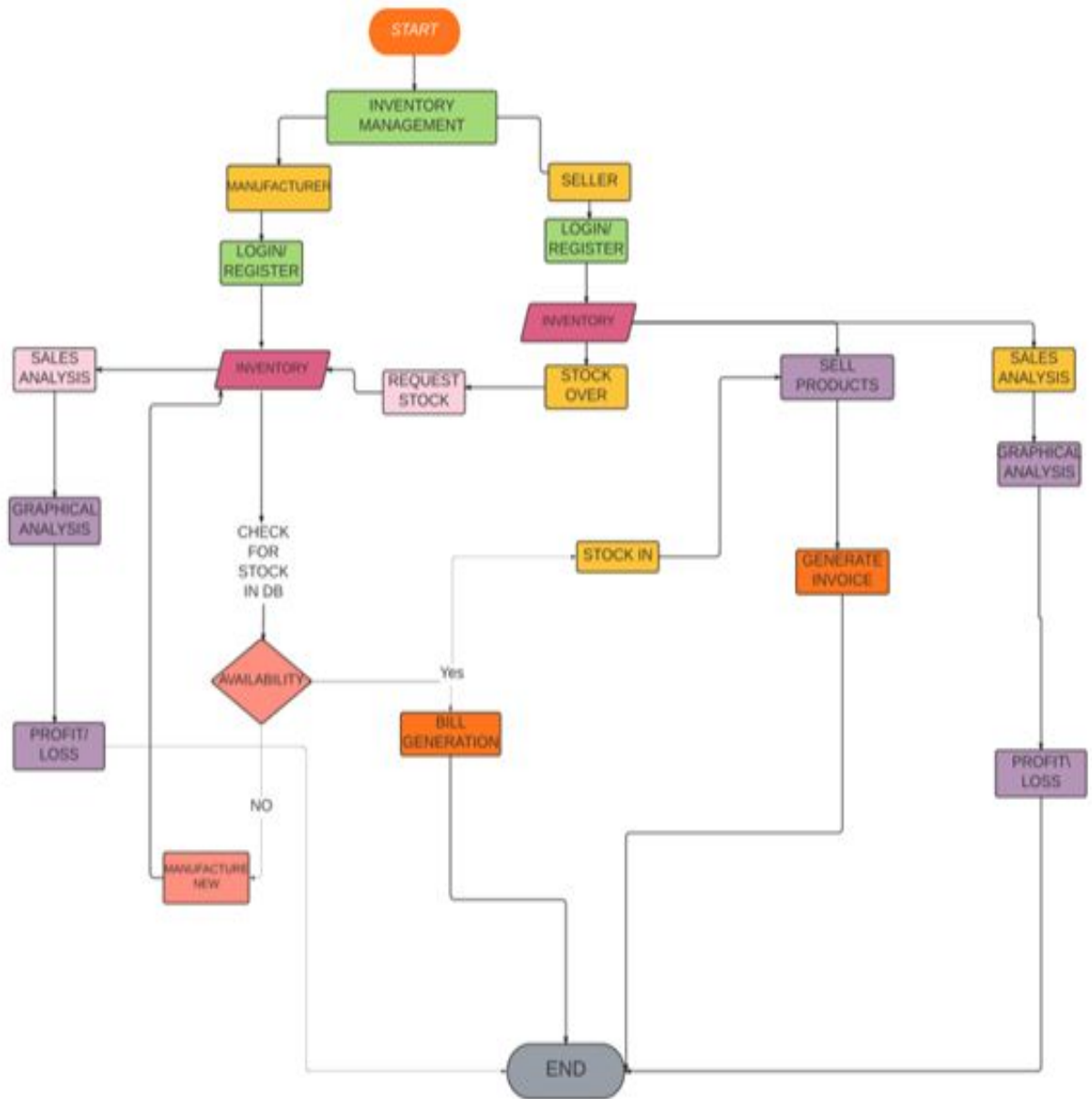
MIDDLEWARE:

Middleware provides a convenient mechanism for filtering HTTP requests entering your application. For example, Laravel includes a middleware that verifies the user of your application is authenticated. If the user is not authenticated, the middleware will redirect the user to the login screen. However, if the user is authenticated, the middleware will allow the request to proceed further into the application.

Additional middleware can be written to perform a variety of tasks besides authentication. A CORS middleware might be responsible for adding the proper headers to all responses leaving your application. A logging middleware might log all incoming requests to your application.

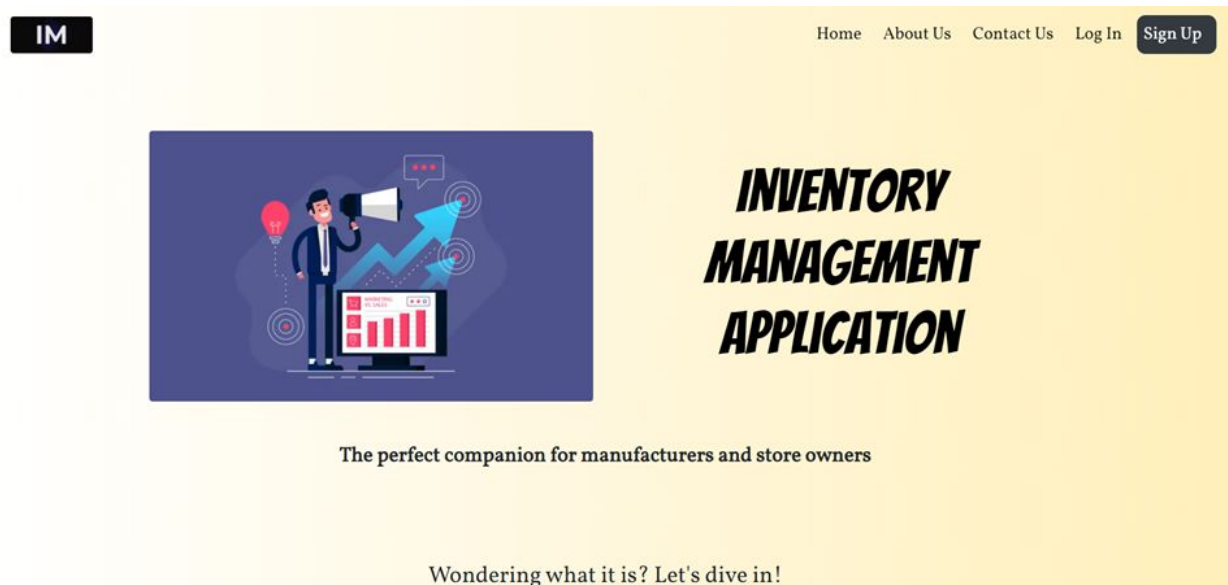
There are several middleware included in the Laravel framework, including middleware for authentication and CSRF protection. All of these middleware are located in the `app/Http/Middleware` directory.

FLOWCHART:

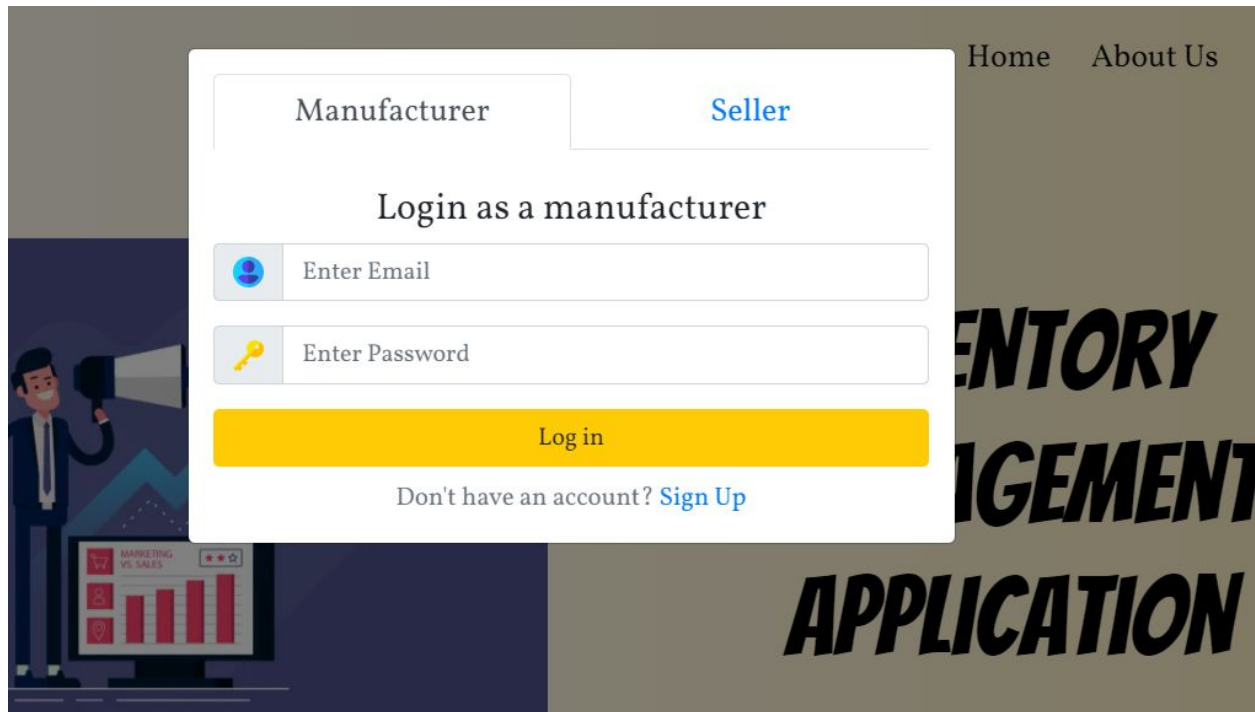


RESULTS AND DISCUSSIONS:

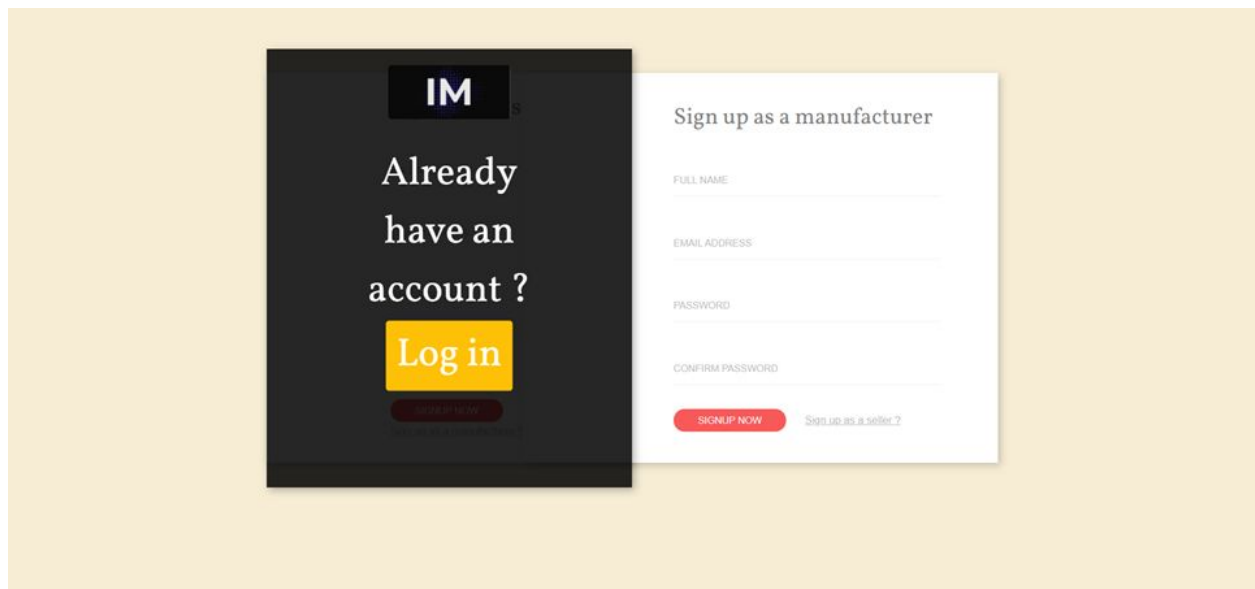
LANDING PAGE:



SIGN-IN PAGE:



SIGN-UP PAGE:



DASHBOARD:



INVENTORY:

MANUFACTURERS INVENTORY:



- It contains details about the products that are available with the manufacturer .
- There is a pop-up displayed below the items listed which contains the requests sent by the seller .Manufacturer can decide whether to grant the request or reject the offer by clicking the button provided.

Manufacturers Inventory

IM

[Home](#) [Sales Analysis](#) [Invoice](#) [Log out](#)

List of Items

Product ID	Product Name	Production cost	Material cost	Price	Quantity	Date of manufacture	Expiry date	
8	Product-2	1200	700	2400	1	2020-11-15	2020-12-12	 

[Add More](#) 

NEW ITEM REQUESTS!

You received item requests from seller.
Click on the button below to view item requests.

[View Requests](#)



SELLERS INVENTORY:

- It contains the products available at the seller which can be sold to the customer directly
- products which can be requested to manufacturer.

IM

HomeSales AnalysisInvoiceLog out

List of Items

Product ID	Product Name	Price	Quantity	Date of manufacture	Name of manufacturer	Expiry date	
25	Product-5	1000	9	2020-11-18	Manufacturer-4	2021-01-23	 

Add More +

Products available for requests

Name of manufacturer	Product Name	Quantity available	Price	Date of manufacture	Expiry date	Operation
sample user1	sample_product	4	1200	2020-11-08	2020-12-05	<div>REQUEST</div>
Sample User	Product-2	1	2400	2020-11-15	2020-12-12	<div>REQUEST</div>

MANUFACTURERS INVOICE GENERATION:

Bill generation for sellers on buying the products from the manufacturer is done here.

Manufacturers details and product details are mentioned and sellers can print their bill online anywhere.

Manufacturers invoice generation

IM

Home Sales Analysis Inventory Log out

Search...

Details		
Customer Name	Products Sold	Operation
Seller-1	2	GENERATE INVOICE
Seller-2	1	GENERATE INVOICE

Manufacturers Invoice

INVOICE

Store Name

Address

Customer Name: Seller-1

Product Name	Price	Quantity	Amount	Date of Transaction
Product-2	2400	5	12000	2020-11-15
Product-1	2000	1	2000	2020-11-15

Number of Items: 2

Total Quantity: 6
Total Amount: 14000

Thank You! Visit Again!

Print Invoice

SELLERS INVOICE GENERATION:

An Invoice is generated for the products sold to customers from the sellers.

1)Customers need to add their products to their cart. They can add multiple products

- 2)After adding, they need to add their details like name and mobile number
- 3)They are also needed to add their store name
- 4)Remaining things are same as of manufacturers bill generation

Sellers invoice generation

IM

[Home](#)
[Sales Analysis](#)
[Inventory](#)

Log out

Customer Details

Customer Name: Customer-1
 Mobile Number: 9728347823
 Date of Transaction: 2020-11-23

Edit

Delete

Cart

Product Name	Price	Quantity sold	Date of manufacture	Expiry date	Operation
Product-5	2100	5	2020-11-18	2021-01-23	<div style="background-color: #dc3545; color: white; padding: 5px 10px; border-radius: 5px;">REMOVE</div>

GENERATE INVOICE

🔍

List of Items

Product ID	Product Name	Price	Selling price	Quantity	Date of manufacture	Name of manufacturer	Expiry date	Operation
25	Product-5	1000	2100	4	2020-11-18	Manufacturer-4	2021-01-23	<div style="background-color: #28a745; color: white; padding: 5px 10px; border-radius: 5px;">Add to Cart</div>
26	Product-2	2400	3200	2	2020-11-15	Sample User	2020-12-12	<div style="background-color: #28a745; color: white; padding: 5px 10px; border-radius: 5px;">Add to Cart</div>

Sellers Invoice

INVOICE

Store Name	Date: 2020-11-16		
Address			
Customer Name: Customer-1	Mobile Number: 9728347823		
Product Name	Price	Quantity	Amount
Product-1	2100	1	2100
Product-2	3200	3	9600
Number of Items: 2		Total Quantity: 4	Total Amount: 11700
Thank You! Visit Again!			
Print Invoice			

SALES ANALYSIS:

A company can analyze their sales in order to improve their “Days In Inventory” number. They can do this by seeing which products sell the fastest and which products take a long time to sell. They can then deem poor-selling products obsolete and abandon them from their inventory. This is inventory control based on sales analysis.

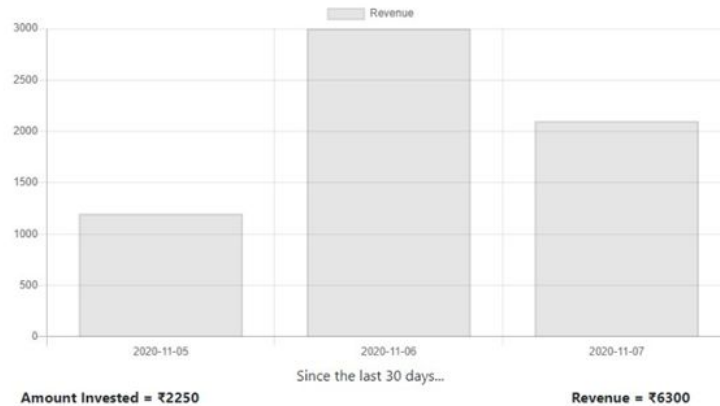
With a set of products that sell well and through eliminating the ineffective products from their inventory, owners are potentially increasing the value of the business. This is both in terms of potential profit from sales and the intrinsic value of the business. When the profitability increases, so does the intrinsic value.

All this is only possible through effective sales analysis. There are a number of inventory and sales analysis tools that can help. In order for companies to succeed and grow their business, an online inventory management system based on sales analysis is a key factor. Without it, while they might still sell stock, they would be nowhere near as efficient.

Users are provided with the data of sales Profit/Loss/Breakeven for their investment

Sales Analysis

PROFIT

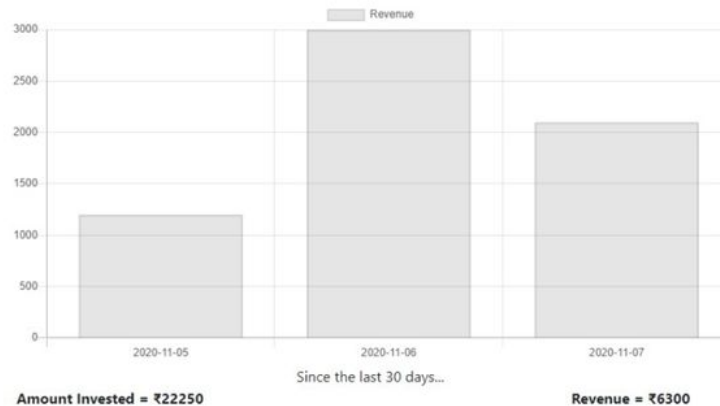


Net Profit Margin: 64.3 %

Your establishment is doing great!

Sales Analysis

LOSS



You need to earn more 15950 rupees to breakeven

Sales Analysis

BREAKEVEN



You made the amount you have invested!

INSIGHTS:

Insight-driven inventory management or stock-control lets businesses move on from the reactive processes of the past and become forward looking and proactive. This approach can result in substantial reductions in cost of goods sold (COGS) and a marked increase in gross margin return on inventory (GMROI).

Insights makes remarkable changes in the development of a store as they remind us of best selling products, Least Selling products, Purchase frequency of customers , Total number of customers, High revenue earned and Least revenue earned etc.

Insights

Insights

Most Selling Product

The most selling product in your store is P3. It has sold 8 units since the last 30 days!
Woah! You might want to invest more on that product!

Least Selling Product

The least selling product in your store is P1. It has sold 1 units since the last 30 days.
Looks like your customers are not interested in that product. Invest less on that.

Purchase Frequency

On an average, your customers buy 3 products on every visit.

Total Number Of Customers

Since the last 30 days, 4 customers have visited your store.

Highest Revenue Earned

The highest revenue that your store has earned is on 2020-11-08
Your store made ₹ 800

Least Revenue Earned

Least revenue that your store has earned is on 2020-11-05
Your store made ₹ 100

TRANSACTION HISTORY:

Inventory transaction history provides a record of all the transactions performed in your company.

For a given range of dates, a person can know the number of transactions made by the users.

Transaction History

IM

Home Back

Log Out

Transaction History

From Date

mm/dd/yyyy

To Date

mm/dd/yyyy

Submit

Product Name	Quantity Sold	Price	Date of Manufacture	Expiry Date	Seller's Email ID	Date of Transaction
Product-2	5	2400	2020-11-15	2020-12-12	seller1@gmail.com	2020-11-14
Product-1	1	2000	2020-11-06	2020-12-07	seller1@gmail.com	2020-11-15
Product-2	2	2400	2020-11-15	2020-12-12	seller2@gmail.com	2020-11-15

FUTURE WORK:

Since this project was started with very little knowledge about the Inventory Management System, we came to know about the enhancement capability during the 37 process of building it. Some of the scope we can increase for the betterment and effectiveness oar listed below:

- Manage Stock Godown wise.
- Use of Oracle as its database.
- Online payment systems can be added.
- Making the system flexible in any type.
- Sales and purchase return systems will be added in order to make return of products.
- Lost and breakage

CONCLUSION:

After the study, we came to a conclusion that, effectiveness of inventory management should improve in all aspects, hence the industry can still strengthen its position by looking into the following.

- The inventory should be fast moving so that warehouse cost can be reduced.
- The finished goods have to be dispatched in feasible time as soon as manufacturing is completed.
- Optimum order quantity should be maintained, hence cost can be minimized.
- Proper inventory control techniques are employed by the inventory control organization within the framework of one of the basic models like ABC, HML and VED etc.

REFERENCES :

<https://laravel.com/docs/8.x/readme>

<https://charts.erik.cat/>

https://www.academia.edu/26003928/Final_Year_Project_On_Inventory_Management_System_Submitted_By