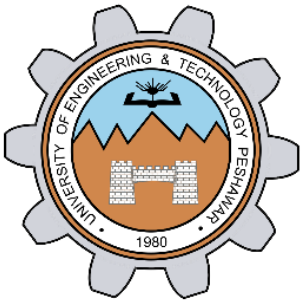
DBMS PROJECT REPORT

ONLINE INVENTORY MANAGEMENT SYSTEM

Key Milestone 04



CSE403L Database Management System Lab

Group members

Arsalan khan (22PWCSE2110)

Waseem (22PWCSE2179)

Adnan Zeb (22PWCSE2191)

Class Section: A

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature:

Submitted to:
Engr.Sumayyea Salahuddin
(JULY 08, 2025)

Department of Computer Systems Engineering
University of Engineering and Technology, Peshawar

Project Title

Online Inventory Management System using Laravel 12 and MySQL

Abstract

This project is a web-based inventory and order management system built with Laravel 12 (PHP framework) and MySQL. It allows users to manage products, suppliers, customers, purchases, and sales orders with real-time stock updates and user-based access. The system helps small and medium businesses maintain clear records, generate reports, and ensure data integrity.

Introduction

Managing inventory and sales manually can be time-consuming and error-prone. This system aims to automate those tasks using a modern PHP framework (Laravel) and a relational database (MySQL). With responsive design and user-friendly interfaces, the application supports streamlined business operations.

Objectives

- To automate inventory tracking
- To manage suppliers and customers efficiently
- To enable real-time stock updates
- To generate purchase and order reports
- To deploy a secure, scalable web system

Modules

- User Authentication (Login/Register)
- Supplier Management
- Customer Management
- Product Management (with Categories & Units)
- Purchases and Purchase Details
- Orders and Order Details
- Stock Updates
- Payment Tracking (paid, due)
- Admin Dashboard with Charts

Tools and Technologies

- Laravel 12 – PHP web framework
- PHP 8.3 – Server-side language (used on InfinityFree)
- MySQL – Relational database
- XAMPP / Laragon – Local development server
- GitHub – Version control
- InfinityFree – Free hosting with PHP 8.3 support

Project Benefits

- Centralized inventory and sales tracking
- Automatic stock updates on purchases and sales
- Reduced risk of overstocking or running out of stock
- Faster customer order processing
- Transparent reporting and accountability
- Easy to maintain and extend

Security Features

- Laravel's built-in CSRF protection
- Hashed passwords using Bcrypt
- Authenticated routes
- Form validation for all input fields

Challenges Faced

- Handling many-to-many relationships with pivot tables (e.g., order_details)
- Real-time stock updates during simultaneous orders
- Deployment limitations with shared/free hosting (e.g., InfinityFree's file limits)

Real-World Use Case

Consider a retail store with multiple products and frequent purchase and sales transactions. Using this system, the store owner can easily:

- Add or update product details
- Track which supplier delivered which product and when
- Generate invoices and receipts for customer orders
- Monitor daily stock movement without relying on manual registers or Excel

System Architecture

The application follows the MVC (Model-View-Controller) architecture:

- Model: Handles data logic and communicates with the MySQL database
- View: Blade templates render dynamic HTML pages
- Controller: Manages user requests and responses

Scalability and Maintenance

Laravel's modular structure and use of Composer packages make the system highly scalable. As the business grows, features like multi-user roles, warehouse management, or product barcodes can be added with minimal code changes.

Data Accuracy

One of the key goals of this system is to ensure that the inventory is always accurate. All stock increases or decreases are logged through purchases and orders, minimizing manual errors and providing a clear audit trail.

Who Can Use This System

- Small businesses (retailers, wholesalers)
- Inventory-based startups
- Schools/colleges managing store inventory
- Office supply departments
- Freelance developers as a base project

System Modules

- User Authentication
- Product Management
- Supplier and Customer Management
- Purchase Module
- Sales and Orders Module
- Stock Control
- Reports and Analytics Dashboard

Entity Analysis Table

Entity Name	Purpose	Key Attributes	Relationships
Users	Manages the system	id, name, email, username, password	One-to-many with Purchases (created_by, updated_by)
Suppliers	Source of purchased products	id, name, email, phone, shopname, bank details	One-to-many with Purchases
Customers	Places orders	id, name, email, phone, address	One-to-many with Orders
Categories	Groups products	id, category_name	One-to-many with Products
Units	Defines measurement units	id, unit_name	One-to-many with Products
Products	Inventory items	id, name, code, buying_price, selling_price, stock	Belongs to Category & Unit; Many-to-many with Orders & Purchases
Purchases	Records supplier purchases	id, purchase_no, purchase_date, supplier_id	Many Purchase_Details; Linked to Supplier & User
Purchase_Details	Line items for purchases	purchase_id, product_id, quantity, unitcost, total	Links Purchases and Products (composite key)
Orders	Records customer orders	id, customer_id, order_date, total, payment_type	Many Order_Details; Linked to Customers
Order_Details	Line items for orders	order_id, product_id, quantity, unitcost, total	Links Orders and Products (composite key)

Entity Explanation with Attributes and Relationships

1. Users

- **Purpose:** System administrators or staff who log in and manage the system.
- **Attributes:** id, name, email, username, password, photo
- **Relationships:** One-to-many with purchases (created_by, updated_by)

2. Suppliers

- **Purpose:** External vendors from whom products are purchased.
- **Attributes:** id, name, email, phone, address, shopname, bank details

- **Relationships:** One-to-many with purchases

3. Customers

- **Purpose:** Buyers who place orders in the system.
- **Attributes:** id, name, email, phone, address, payment details
- **Relationships:** One-to-many with orders

4. Categories

- **Purpose:** Logical grouping of products.
- **Attributes:** id, category_name
- **Relationships:** One-to-many with products

5. Units

- **Purpose:** Measurement units for products (e.g., kg, pcs).
- **Attributes:** id, unit_name
- **Relationships:** One-to-many with products

6. Products

- **Purpose:** Items available for purchase and sale.
- **Attributes:** id, name, code, buying price, selling price, stock, image
- **Relationships:**
    - Belongs to a category
    - Belongs to a unit
    - Many-to-many with purchases (via purchase_details)
    - Many-to-many with orders (via order_details)

7. Purchases

- **Purpose:** Records of products bought from suppliers.
- **Attributes:** id, purchase_no, purchase_date, status, supplier_id
- **Relationships:**
    - Many purchase_details
    - Linked to suppliers and users

8. Purchase_Details

- **Purpose:** Line items for each purchase.
- **Attributes:** purchase_id, product_id, quantity, unitcost, total

- **Relationships:**
    - Composite key
    - Connects purchases and products

9. Orders

- **Purpose:** Sales made to customers.
- **Attributes:** id, order_date, total, status, customer_id, invoice_no, payment_type
- **Relationships:**
    - Many order_details
    - Linked to customers

10. Order_Details

- **Purpose:** Line items for each order.
- **Attributes:** order_id, product_id, quantity, unitcost, total
- **Relationships:**
    - Composite key
    - Connects orders and products

Entity Relationship Diagram (ERD)

1. users

- `id` (PK)
- name
- email
- username
- photo
- password

2. suppliers

- `id` (PK)
- name
- email
- phone
- address
- shopname
- type

- bank_name
- account_holder
- account_number
- photo

3. customers

- `id` (PK)
- name
- email
- phone
- address
- type
- bank_name
- account_holder
- account_number
- photo

4. categories

- `id` (PK)
- category_name

5. units

- `id` (PK)
- unit_name

6. products

- `id` (PK)
- product_name
- product_code
- buying_price
- selling_price
- stock
- product_image
- `category_id` (FK) → categories.id
- `unit_id` (FK) → units.id

7. purchases

- `id` (PK)
- purchase_date

- purchase_no
- purchase_status
- `supplier_id` (FK) → suppliers.id
- `created_by` (FK) → users.id
- `updated_by` (FK) → users.id

8. purchase_details

- `purchase_id` (FK, PK) → purchases.id
- `product_id` (FK, PK) → products.id
- quantity
- unitcost
- total

9. orders

- `id` (PK)
- order_date
- order_status
- total_products
- sub_total
- vat
- total
- invoice_no
- payment_type
- pay_int
- due_int
- `customer_id` (FK) → customers.id

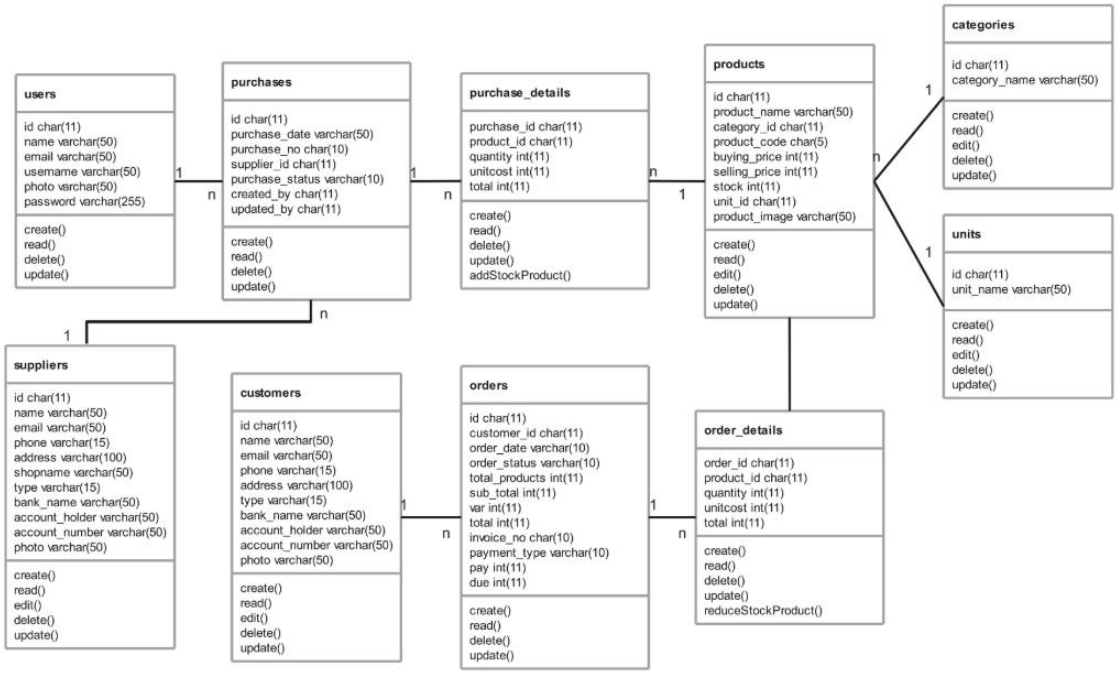
10. order_details

- `order_id` (FK, PK) → orders.id
- `product_id` (FK, PK) → products.id
- quantity
- unitcost
- total

Relationships

- **One-to-Many:**
    - users → purchases
    - suppliers → purchases
    - customers → orders

- orders → order_details
- purchases → purchase_details
- products → purchase_details / order_details
- categories → products
- units → products

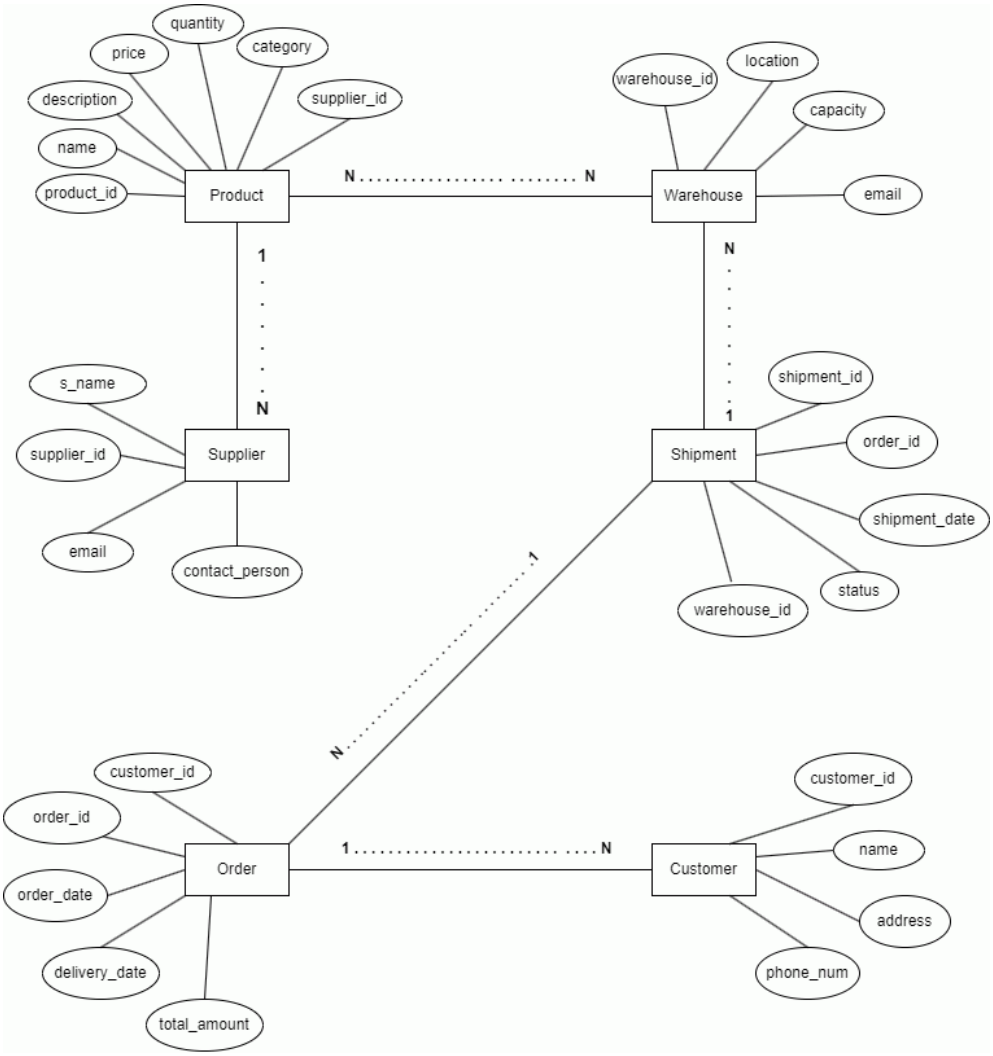


Enhanced Entity Relationship Diagram (EERD)

The Enhanced Entity Relationship Diagram (EERD) expands on the basic ERD by adding more advanced modeling features such as specialization, generalization, and more detailed relationships. It is used to better represent real-world scenarios in complex systems like inventory management.

In this project:

- **Customers** and **Suppliers** are generalized as **Persons**, since both share common attributes (e.g., name, email, phone).
- **Products** are associated with both **Categories** and **Units**, showing classification and measurement relationships.
- **Purchases** and **Orders** are transactional entities connecting **Users**, **Suppliers**, **Customers**, and **Products**.
- **Purchase_Details** and **Order_Details** are used to handle many-to-many relationships with quantities and prices.



Laravel Development Process

1: Setup Project

composer create-project laravel/laravel inventory-system
cd inventory-system
php artisan serve

2: Configure `.env`

DB_DATABASE=inventory_db
DB_USERNAME=root
DB_PASSWORD=

3: Create Models and Migrations

php artisan make:model Product -m
php artisan make:model Supplier -m
php artisan make:model Order -m

4: Run Migrations

php artisan migrate

5: Create Controllers

php artisan make:controller ProductController
php artisan make:controller SupplierController

6: Blade Templates & Routes

- Views stored in `/resources/views`
- Routes defined in `routes/web.php`

7: Authentication

composer require laravel/ui
php artisan ui bootstrap --auth
npm install && npm run dev

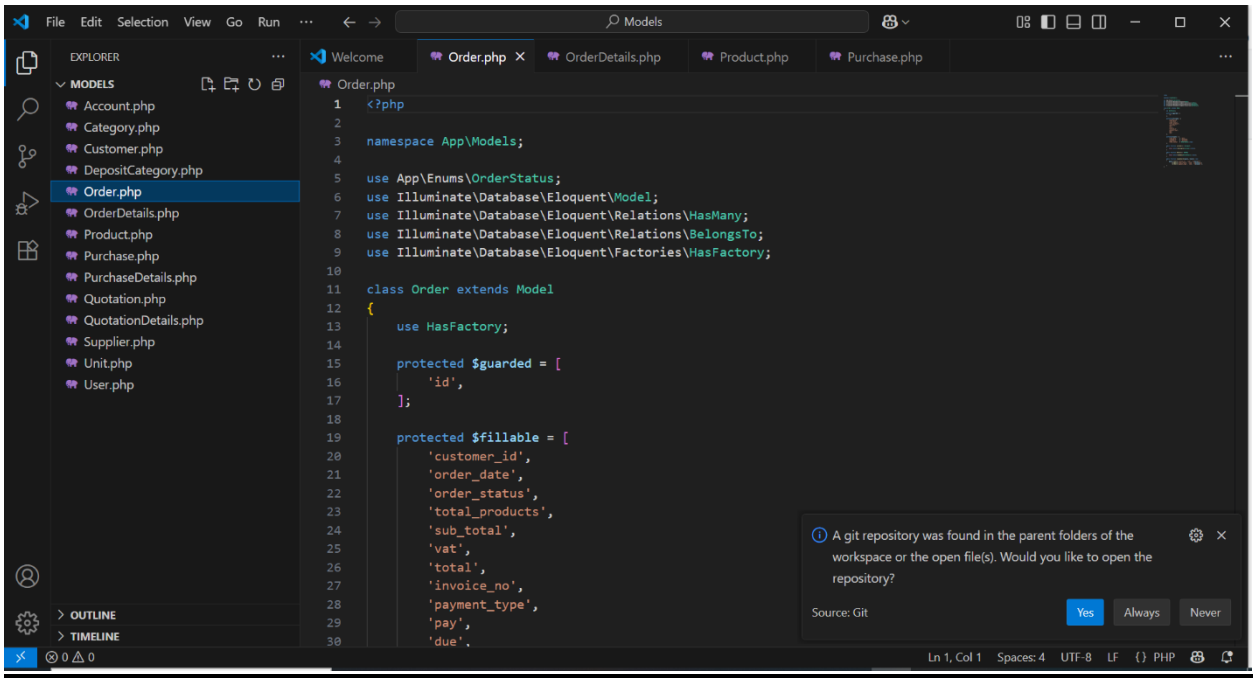
Implementation Details

The system was built using **Laravel 8** and **MySQL**. Laravel's MVC structure was used to organize models, controllers, and views. All modules (products, orders, purchases, users) include full CRUD functionality using Eloquent ORM. Authentication is handled with Laravel UI.

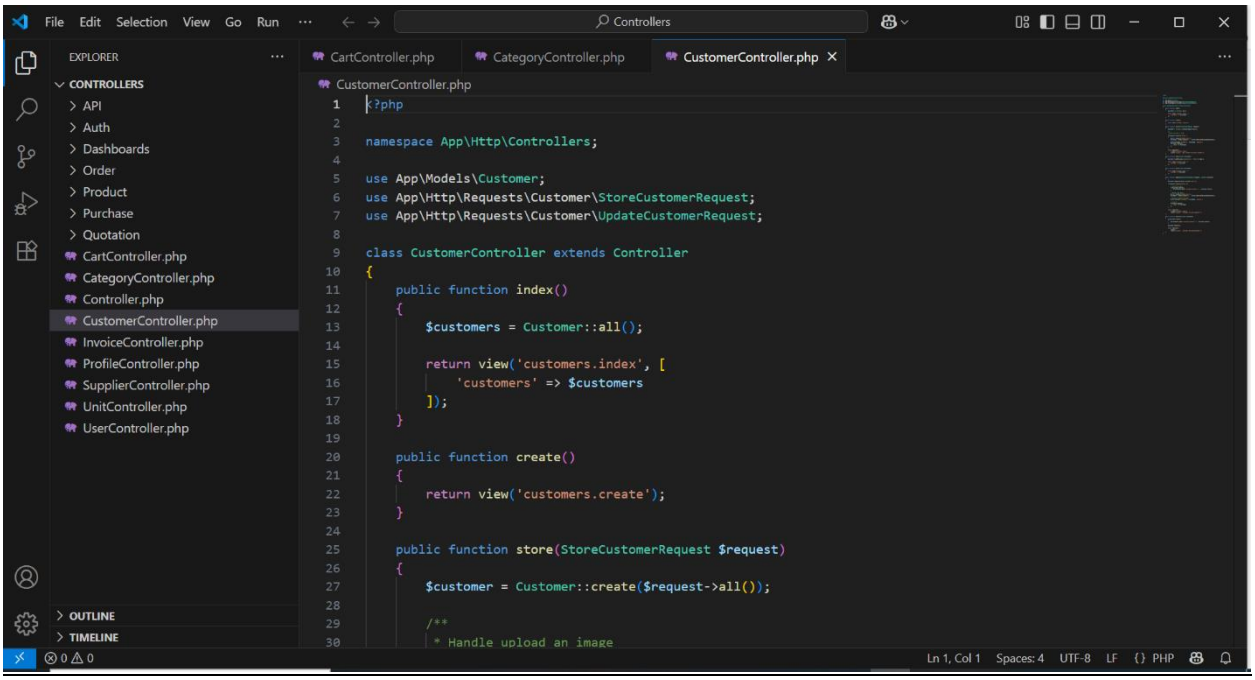
The project was deployed on **InfinityFree** using **PHP 8.3**, and the database was imported via phpMyAdmin.

LARAVEL MVC IMPLEMENTATION

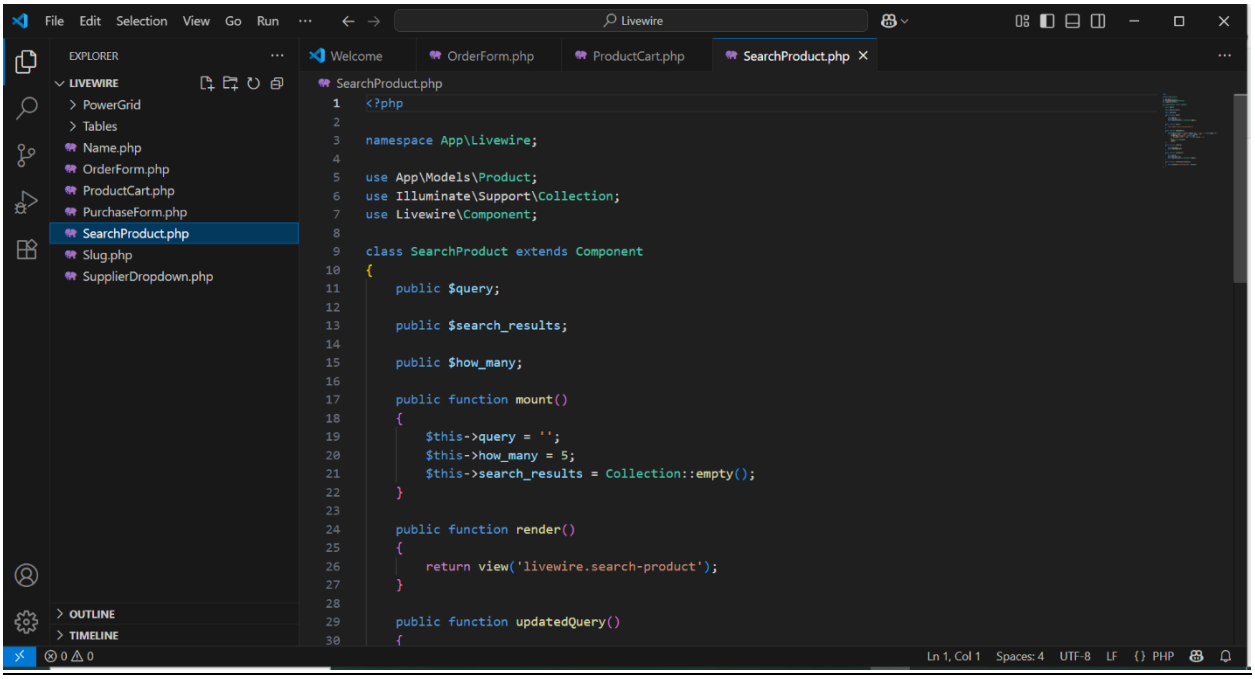
MODEL LAYER



CONTROLLER LAYER



LIVEWIRE



Cloud MySQL Database Deployment

Steps:

1. Sign in to Aiven.io and create a new MySQL service.
2. After successful creation, the following database credentials were provided:

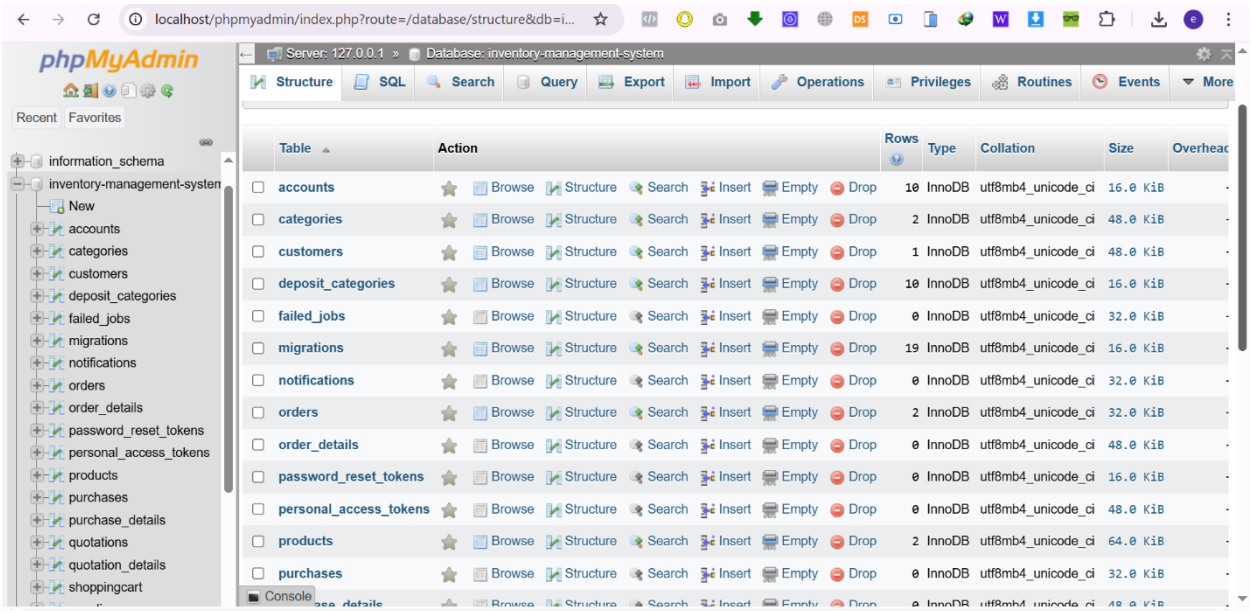
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=inventory-management-system
DB_USERNAME=root
DB_PASSWORD=

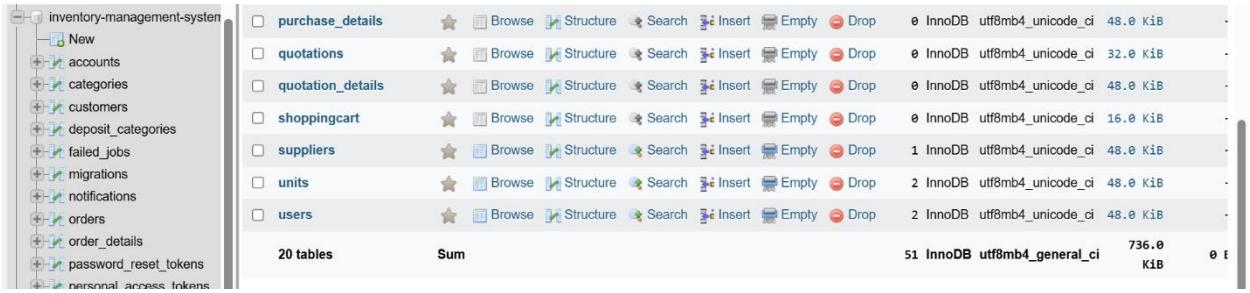
3. These values should be added to Laravel's `.env` file to connect the application to the cloud MySQL database.

MySQL Database

```
MariaDB [(none)]> USE `inventory-management-system`;
Database changed
MariaDB [inventory-management-system]> SHOW TABLES;
+--------------------------------------+
| Tables_in_inventory-management-system |
+--------------------------------------+
| accounts                             |
| categories                           |
| customers                            |
| deposit_categories                   |
| failed_jobs                          |
| migrations                           |
| notifications                        |
| order_details                        |
| orders                               |
| password_reset_tokens                |
| personal_access_tokens               |
| products                             |
| purchase_details                     |
| purchases                            |
| quotation_details                    |
| quotations                           |
| shoppingcart                         |
| suppliers                            |
| units                                |
| users                                |
+--------------------------------------+
20 rows in set (0.003 sec)
```

localhost/phpmyadmin

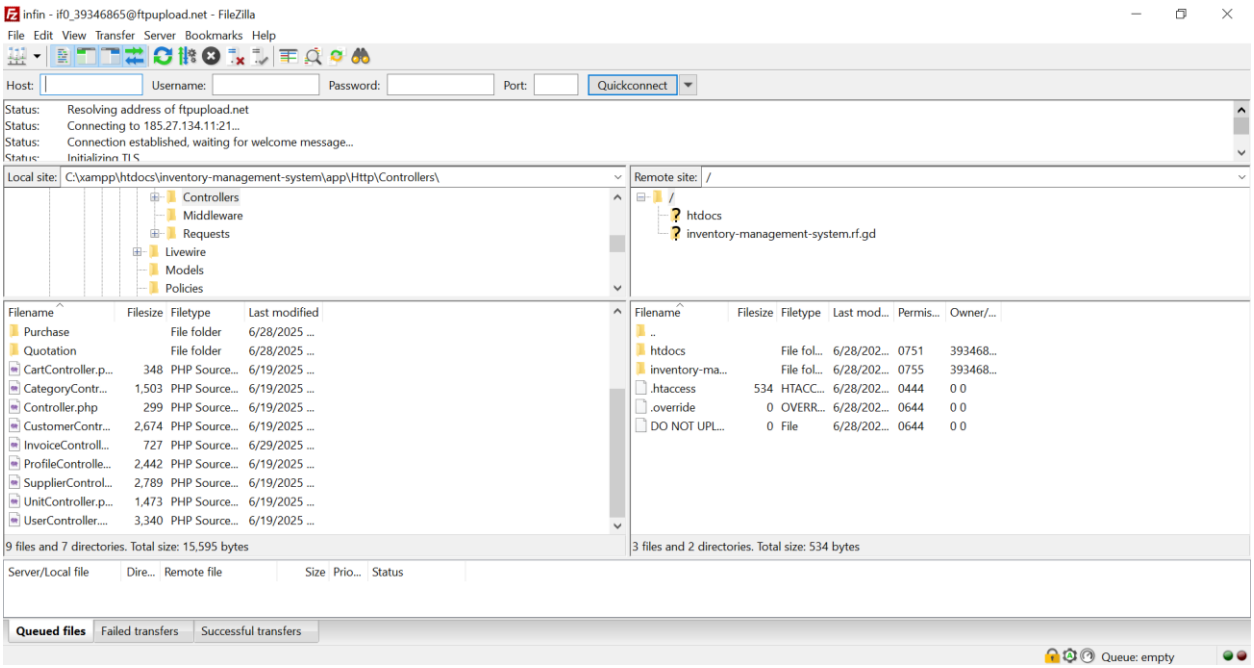




Deployment:

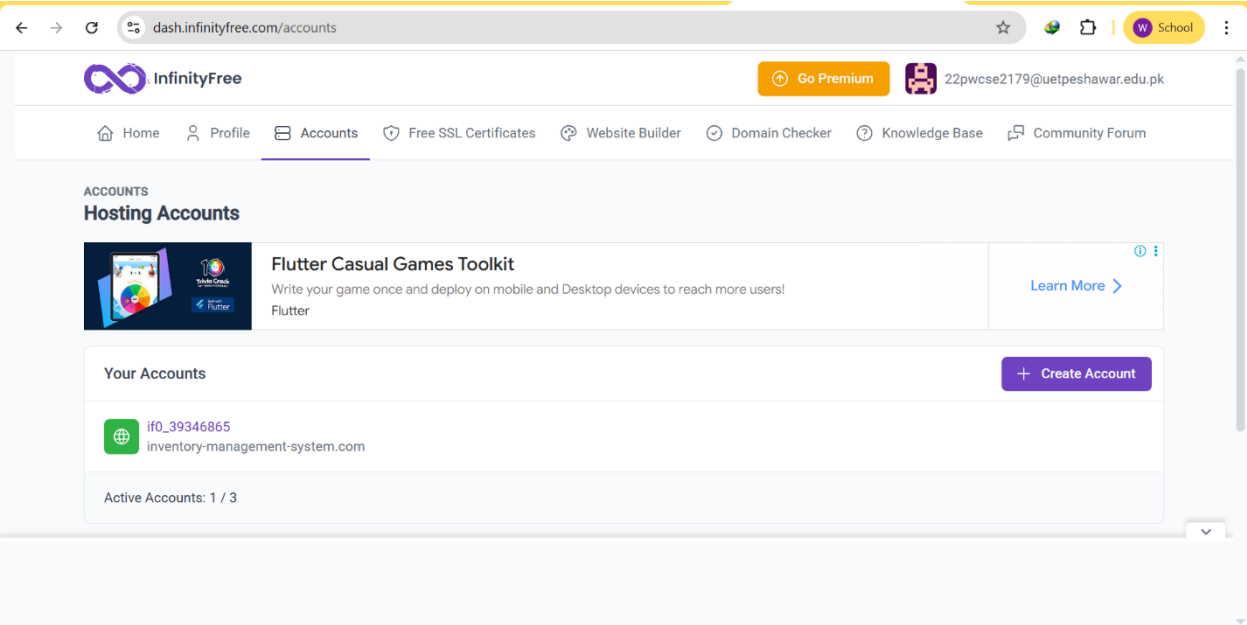
1. Upload Project to FileZilla

Instead of GitHub, the Laravel project was uploaded using **FileZilla** — a file hosting platform that supports direct file uploads.



2. Deploy to InfinityFree

- Create a free hosting account on InfinityFree.net.
- Upload the extracted `public/` folder contents into the `/htdocs` directory via File Manager or FTP (e.g., FileZilla).
- Ensure the main Laravel `index.php` and `.htaccess` are inside `/htdocs`

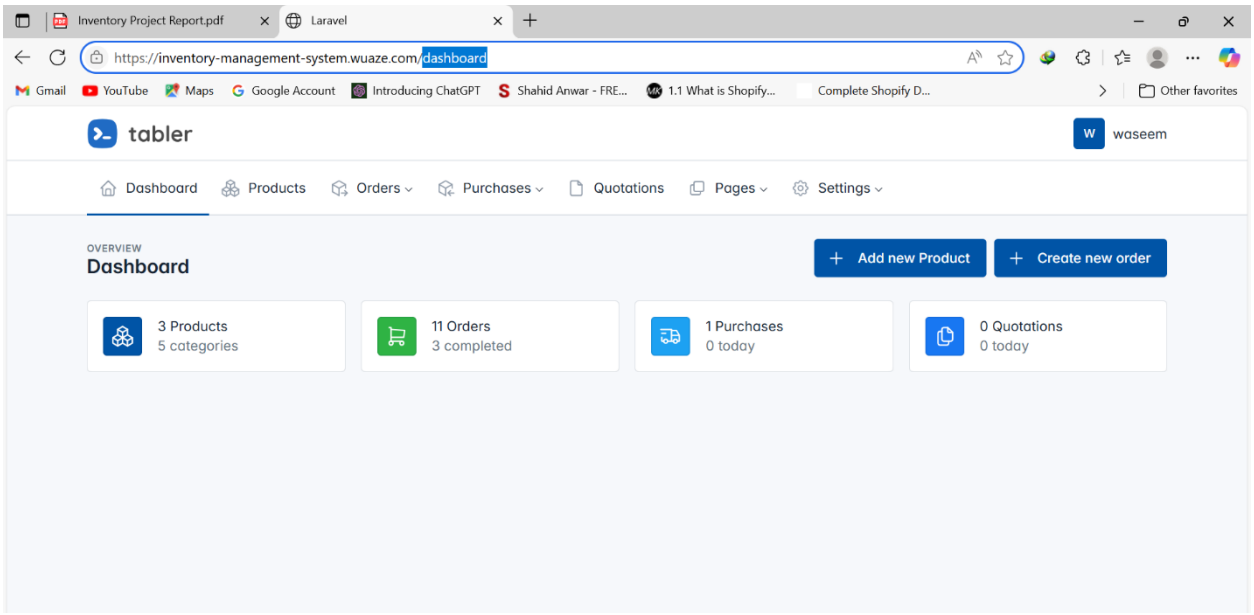


3. Configure `.env` File

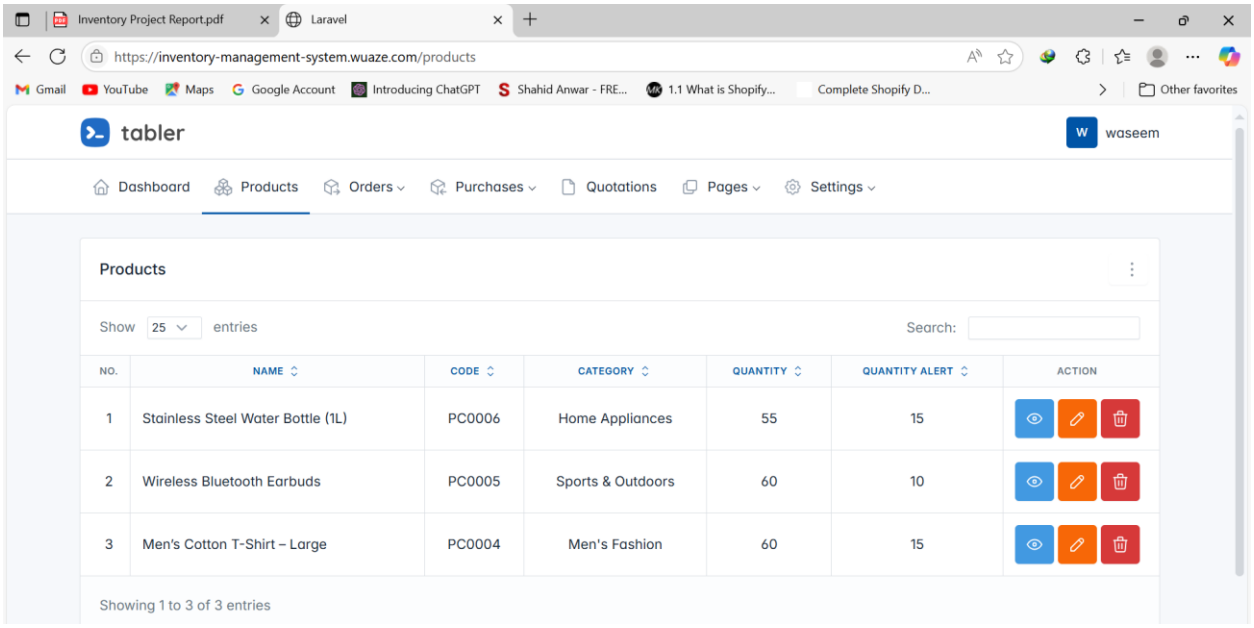
Update the database credentials in your `.env` file using Aiven.io's MySQL service:

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:QZdeSoikDHSgZkiVGVIvRlqaBVcSGN/1JXQOEHKUFcc=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=sql211.infinityfree.com
DB_PORT=3306
DB_DATABASE=if0_39346865_inventory_management_system
DB_USERNAME=if0_39346865
DB_PASSWORD=was744012345

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

```

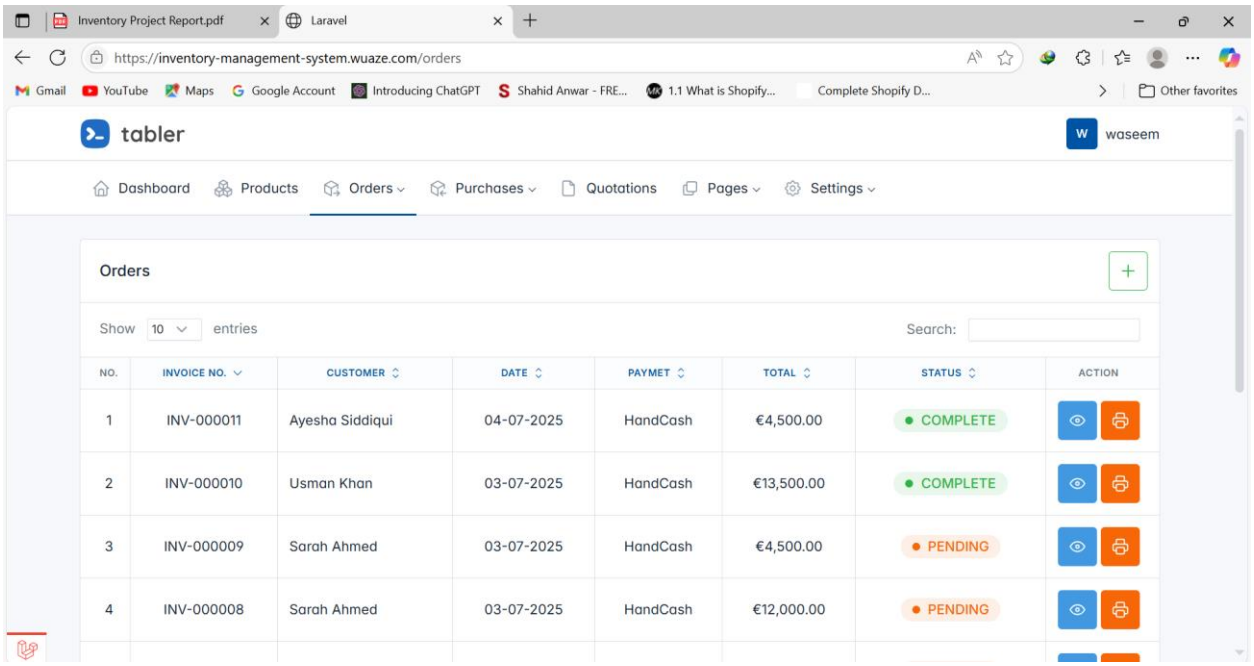
Dashboard



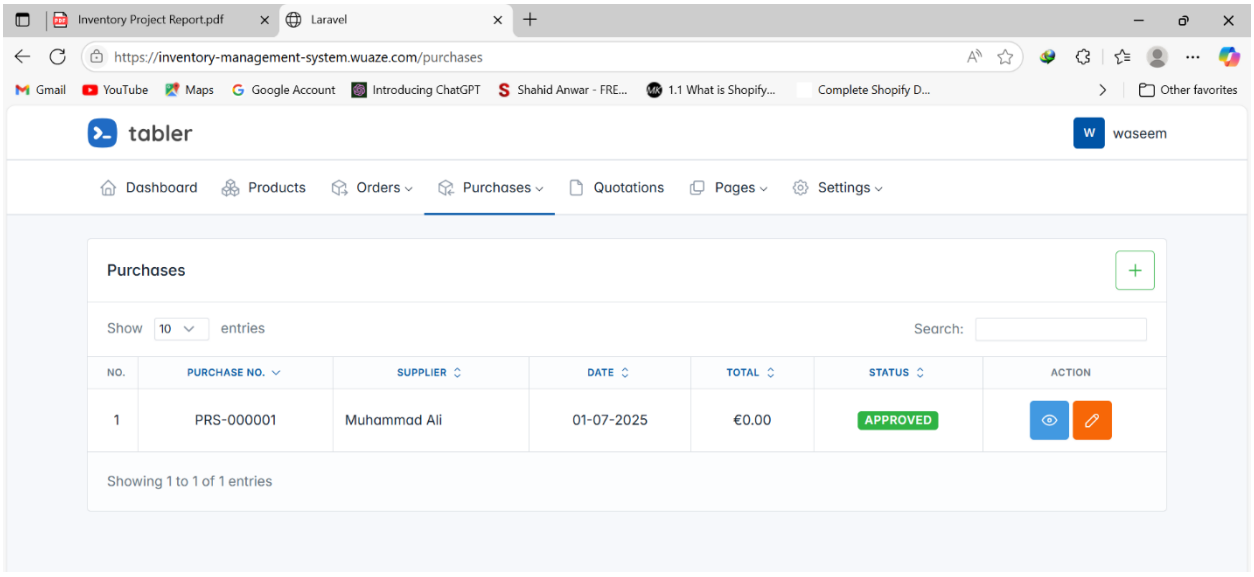
Products Page



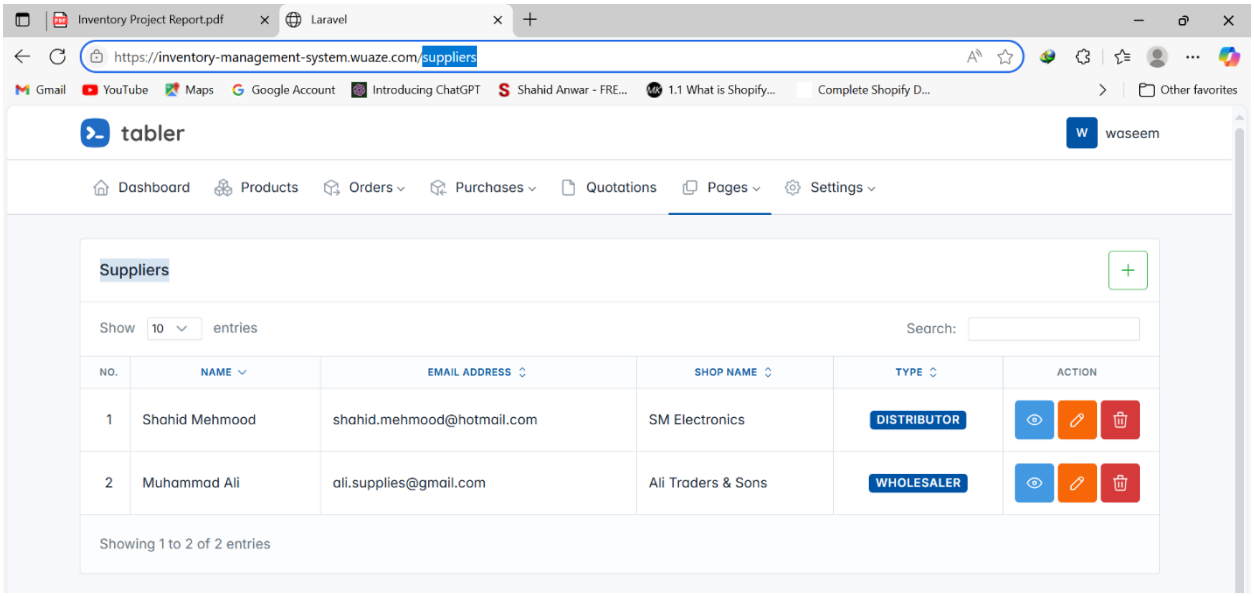
Orders Page



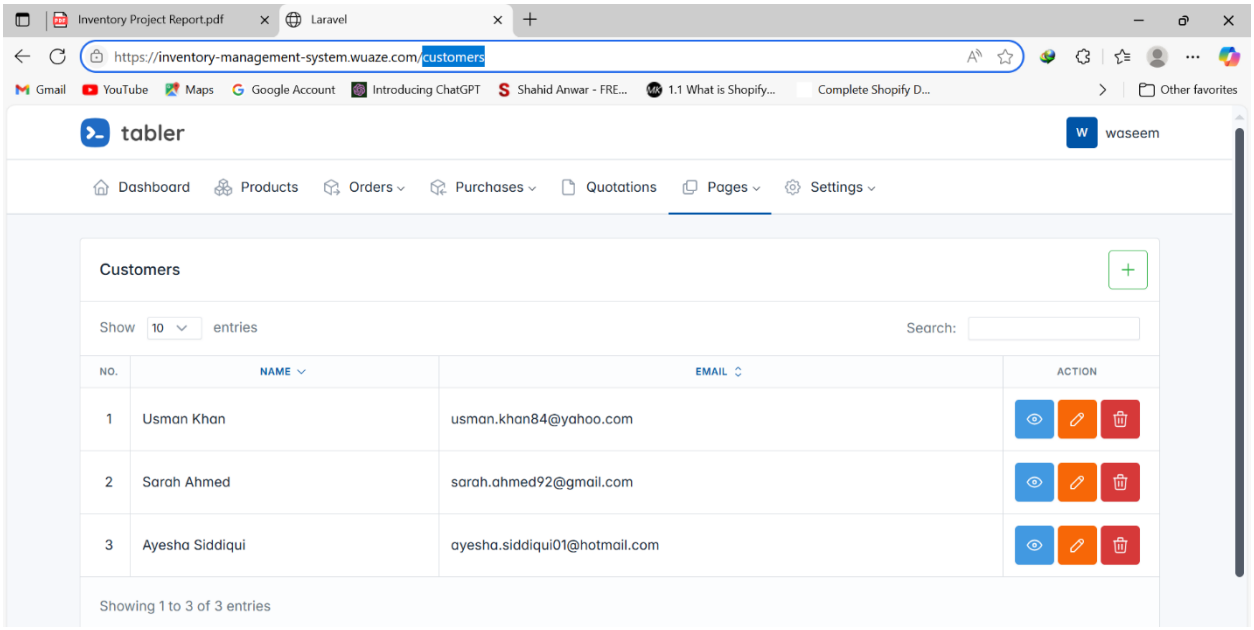
Purchases Page



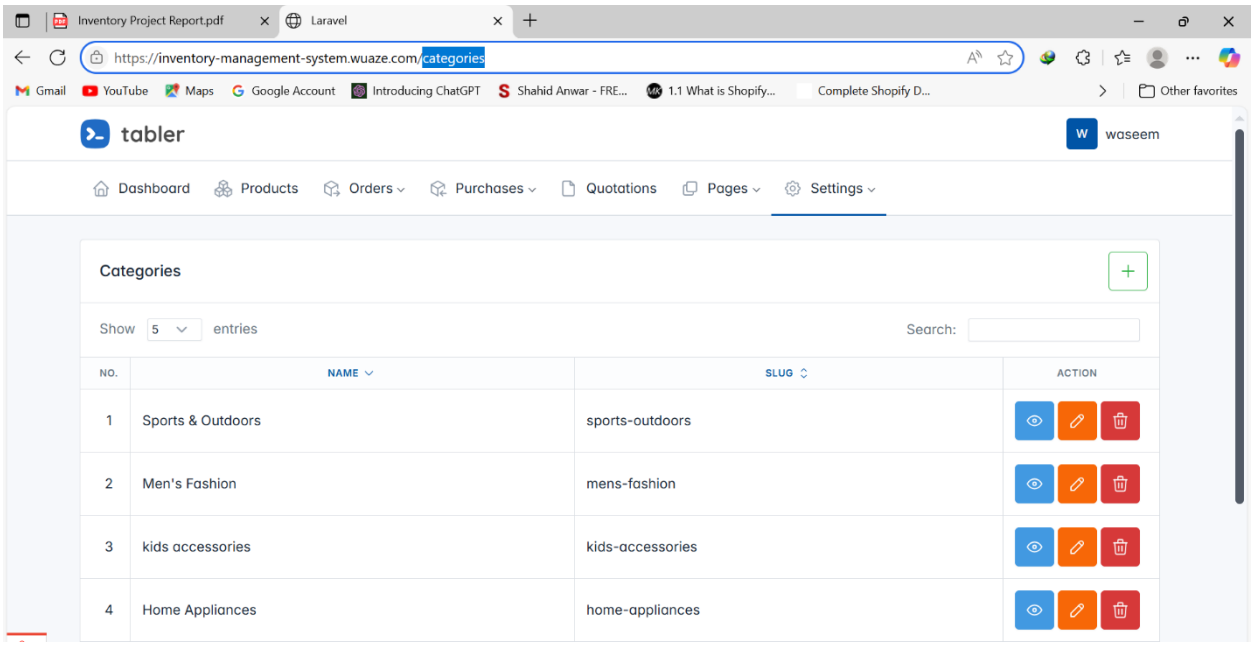
Suppliers



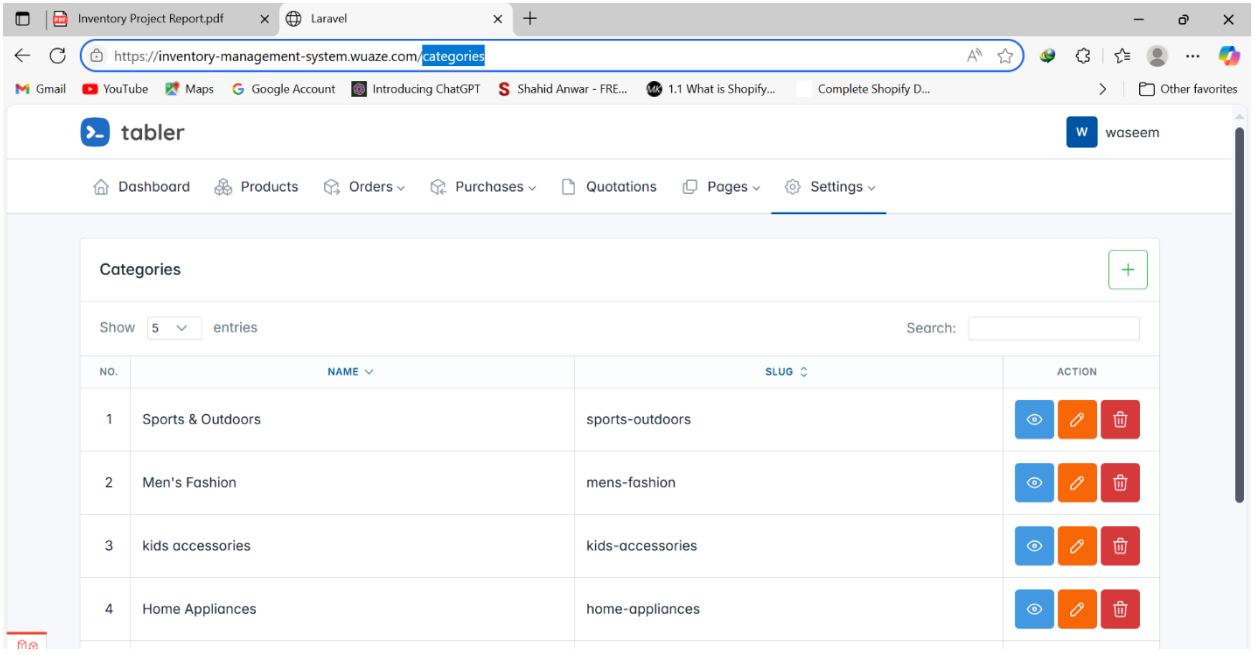
Customers



Categories



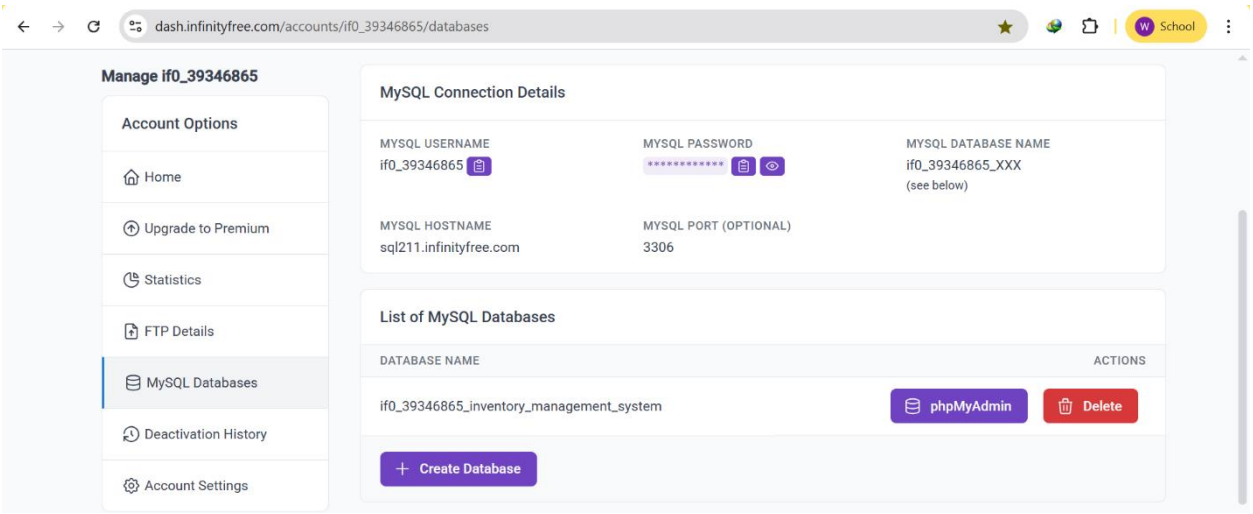
Units

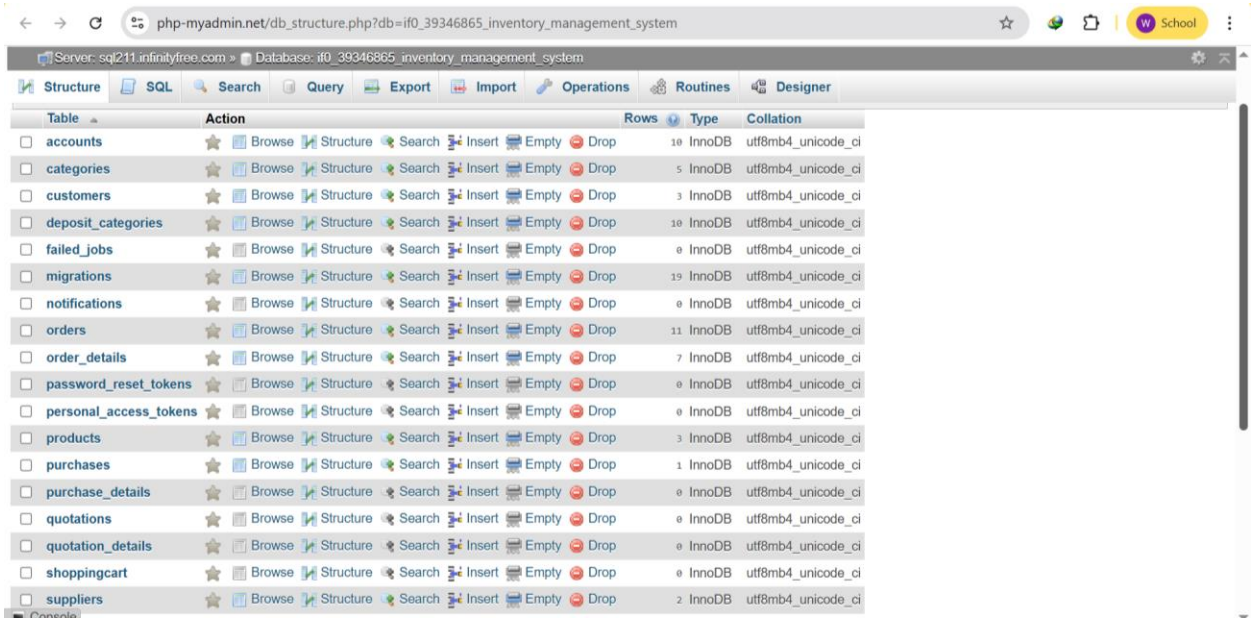


4. Import the Database

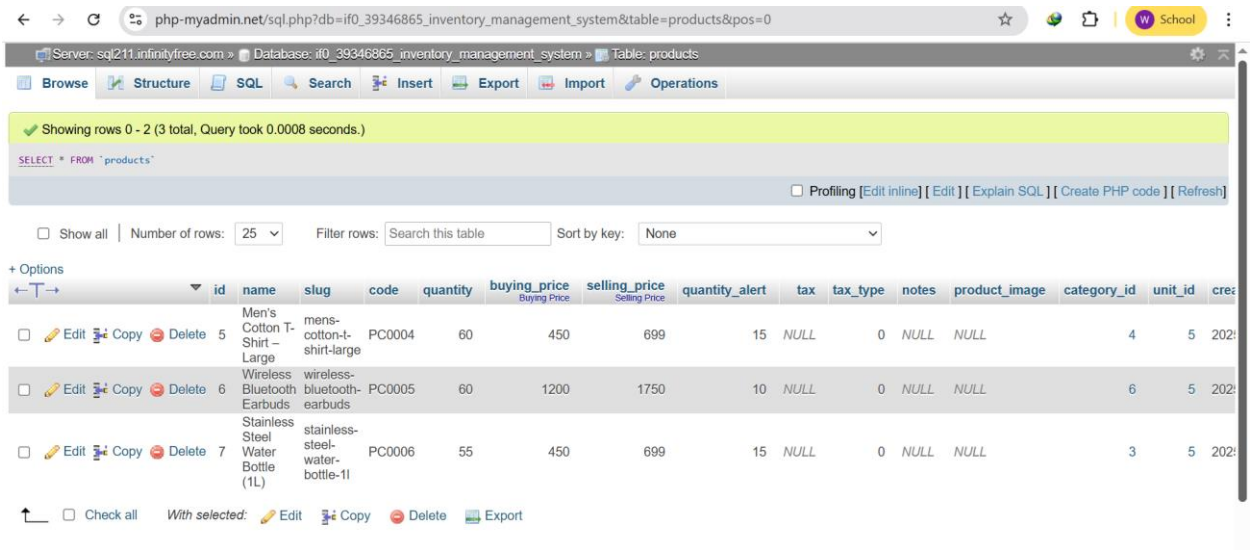
- Go to **phpMyAdmin** in InfinityFree control panel.
- Create a new database (e.g., inventory-management-system).
- Import your `.sql` file using the Import tab.

MySQL Connection Details

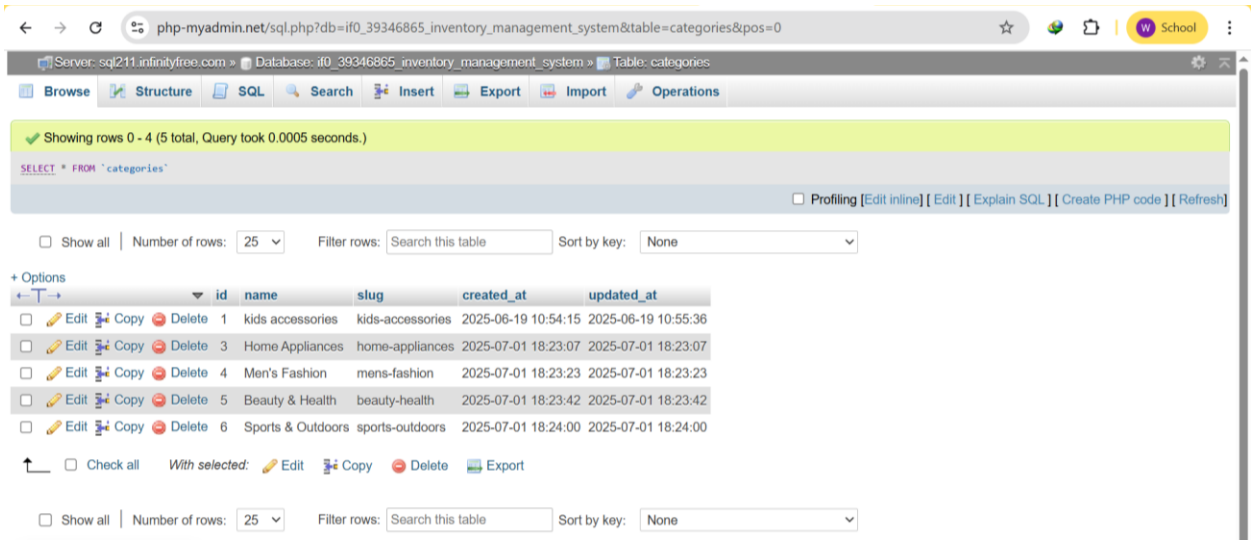




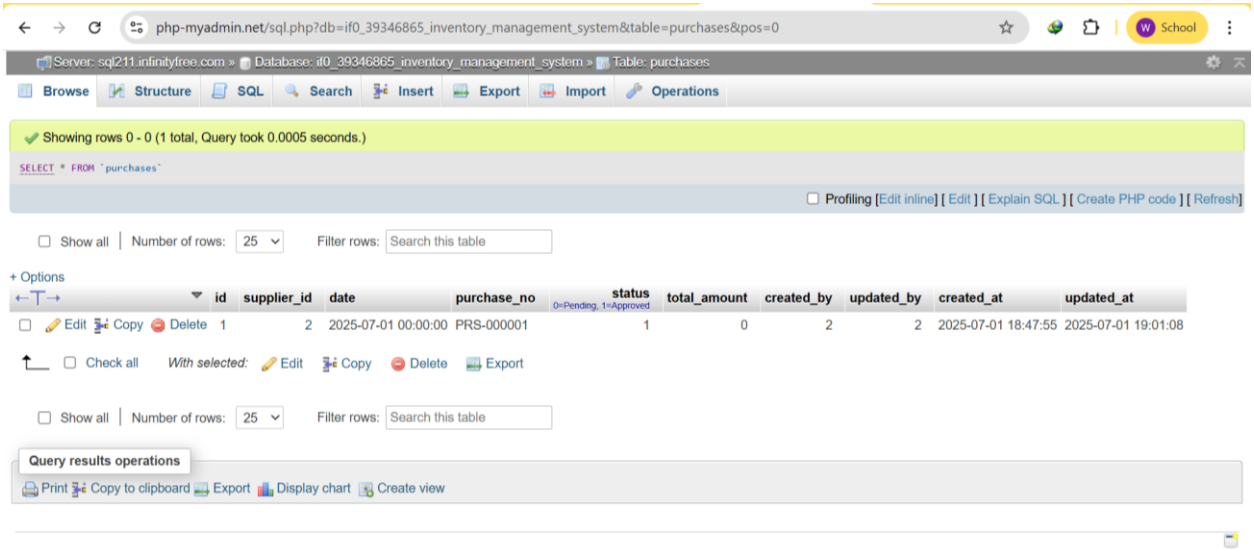
Products



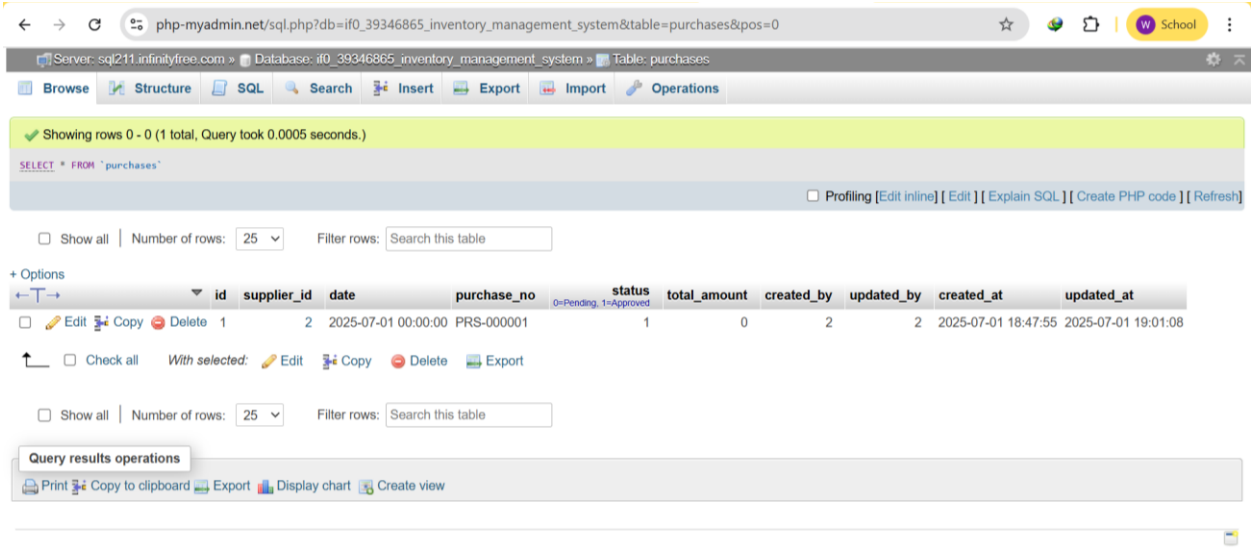
Categories



Purchases



Units



Deployment Links:

Website Link: https://inventory-management-system.wuaze.com/dashboard

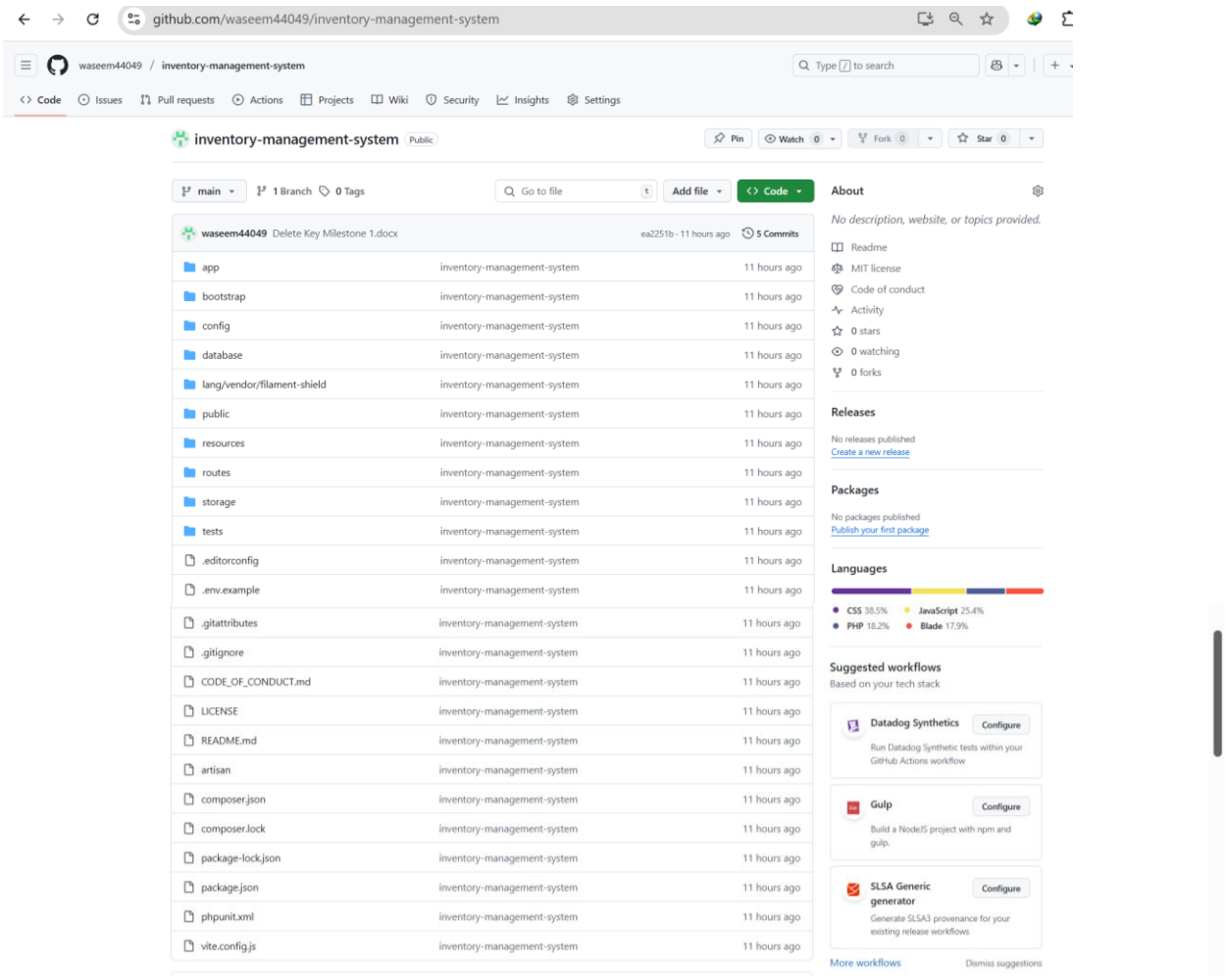
MySQL Database: https://php-myadmin.net/db_structure.php?db=if0_39346865_inventory_management_system

Admin Panel: https://dash.infinityfree.com/accounts/if0_39346865

Admin Credentials: USERNAME if0_39346865 | Password was744012345

Github.com

https://github.com/waseem44049/inventory-management-system



Conclusion

The Inventory Management System was successfully designed and implemented using Laravel 12 and MySQL. It streamlines key business operations such as product management, purchases, sales, and reporting. By using modern web technologies and best practices, the system provides an efficient, user-friendly, and scalable solution for small to medium-sized businesses.

Deployment on **InfinityFree** with a cloud-hosted MySQL database demonstrates the system's flexibility for real-world usage. Overall, this project improves inventory visibility, reduces manual effort, and ensures data accuracy — helping businesses manage their stock and orders effectively.

REFERENCES:

- GITHUB: https://github.com/fajarghifar/inventory-management-system
- YOUTUBE: https://www.youtube.com/watch?v=jk8L4_Wx40U
- DEEPSEEK: https://chat.deepseek.com/a/chat/s/a015c34d-5878-4c89-9cb1-ecd7c1ff4e56
- Chatgpt: https://chatgpt.com/c/686caf0d-040c-8003-a9fc-7925c299c3fc