**🌟 What we have done so far**

1. **EC2 Server Setup**
   * Launched an Ubuntu EC2 instance.
   * Installed **Apache + PHP**.
   * Verified Apache is running (sudo systemctl status apache2).
2. **Website Deployment**
   * Copied your website files (index.php, CSS, JS, images) to /var/www/html/.
   * Set proper **permissions**: www-data:www-data and 755.
   * Restarted Apache to apply changes.
3. **RDS Database**
   * Created an **AWS RDS MySQL instance** (restaurant-db) for storing orders.
   * Configured **security group** to allow EC2 access.
   * Tested the connection using nc -zv ... → **connection successful**.
4. **Database Connection in PHP**
   * Created db\_connect.php with RDS credentials.
   * Tested connection in index.php.
5. **Customer Interface (Front-End)**
   * index.php allows customers to:
     + View menu items
     + Place orders
   * Orders are inserted into the **RDS orders table**.
6. **Admin Dashboard**
   * Created admin.php:
     + Lists all orders
     + Shows customer details, item, price, order time, status
     + Lets admin update status (Completed / Cancelled)
   * Styling applied to make it user-friendly.
   * Connected to the same **RDS database**.
7. **Security**
   * We discussed optional protections:
     + Apache .htaccess for login
     + Only allow admin access to authorized users

**🔜 What we are going to do next**

1. **Protect the admin dashboard**
   * Add **login system** or .htaccess so only the restaurant owner can access.
2. **Deploy globally with a domain and HTTPS**
   * Buy a domain (or use Route 53).
   * Assign domain to EC2.
   * Enable SSL with **Let’s Encrypt** (certbot).
3. **Set up CI/CD pipeline**
   * Use **GitHub Actions** to automatically deploy website updates:
     + Front-end files (index.php, CSS/JS)
     + Admin dashboard (admin.php)
     + Database connection (db\_connect.php)
   * So you don’t have to manually copy files every time.
4. **Optional improvements**
   * Notifications for new orders (email or SMS).
   * Mobile-friendly admin dashboard.
   * Scaling for high traffic (load balancer, multiple EC2s).

✅ **In short:**

* **Customers** can view menu and place orders → stored in RDS.
* **Admin/owner** can see orders in dashboard and update status.
* **Next step**: Make it **secure, global, and automatic** (HTTPS + CI/CD).