## Overview

This project is a custom-built e-commerce web application named SuperLian Tech, designed to offer a user-friendly experience for browsing, filtering, and purchasing mobile devices. It supports both frontend and backend functionalities, including user authentication, cart management, checkout processing, and order history.

## Concept and Motivation

The goal was to build a production-ready, full-stack e-commerce platform with real-world user flows from registration to final order placement. I wanted to simulate a professional online store with dynamic content, secure APIs, and a polished interface, while also learning to deploy such systems using modern DevOps practices.

## Technology Stack

- **Frontend**: React (Vite), Tailwind CSS, React-Redux
- **Backend**: Django, Django RES.T Framework
- **Database**: SQLite
- **Authentication**: Token-based JWT authentication with DRF
- **DevOps/Hosting**: Namecheap, Ubuntu 22.04 VPS, Gunicorn, Nginx, SSL via Let's Encrypt

## Making Of (Technical Process)

- **Cart Logic & Filtering:** Built a cart system that tracks item color/size/quantity dynamically using Redux and Django models; all filter options load based on available product variations.
- **Pagination & API Management:** Pages with listing such as products are paginated using *react-paginate* library. React components re-render based on pagination metadata.
- **Profile & Order Management:** Integrated user profile updates and order history access with secured API endpoints.
- **Custom Admin Role:** Admin users can create, update, and delete products, as well as see orders.
- **Security:** The backend APIs use token-based authentication. Only authenticated users can place orders or access profile data. Admin-only endpoints are protected using Django permissions.

- **Deployment Challenges Solved:**
    - SSL cert integration (resolved issues with .crt and Nginx path errors)
    - Media routing between Django and Nginx
    - Initial media uploads failed due to Django's static/media routing. Solved by correctly configuring Nginx to handle /media/ requests
    - Gunicorn socket failures and permissions

## Lesson Learned

- **Start small**: I started building too many features at once, and it got hard to manage. I backtracked and broke things down into small modules (e.g., filters, cart logic), and that helped significantly.
- **API Performance Matters**: Pagination, query filtering, and caching (on the front end) were essential in keeping the app fast.
- **Front-end libraries save time**: Tailwind allowed me to design components faster without getting bogged down in CSS.
- **Know your deployment workflow**: Setting up Gunicorn + Nginx + SSL on a VPS was more complex than expected. I learned to debug configuration using system logs.
- Pagination state would reset unexpectedly in React. Resolved using ReactPaginate and proper dependency tracking.

## Future Work

- Integrate real payment processing systems such as Stripe or PayPal.
- Implement and add user reviews and ratings for products.
- Add email confirmation and implement notifications for order.
- Improve the login and registration workflow using email confirmation etc.
- Implement search and more robust filtering options.
- Implement AI recommendation system.