NOMAS E-Commerce Platform - Complete Deployment & Setup Guide

Table of Contents

Project Overview

NOMAS is a modern, full-stack e-commerce platform built with:

- **Frontend**: React 19 with Tailwind CSS 4
- **Backend**: Node.js/Express with tRPC
- **Database**: MySQL (or compatible database)
- **Authentication**: Built-in OAuth integration
- **Payment**: Cash on Delivery (COD) and Card payment forms

Key Features

- User authentication and account management

- Product catalog with category filtering
- Shopping cart functionality
- Checkout with multiple payment methods
- Order tracking and management
- Admin dashboard for product and order management
- Responsive design for all devices

System Requirements

For Local Development

- **Node.js**: v18 or higher
- **npm/pnpm**: Latest version
- **MySQL**: v8.0 or higher (or MariaDB 10.5+)
- **Git**: For version control

For Production

- **Server**: Linux (Ubuntu 20.04 LTS or higher recommended)
- **Node.js**: v18 or higher
- **MySQL**: v8.0 or higher
- **Nginx/Apache**: For reverse proxy
- **SSL Certificate**: Let’ s Encrypt (free)
- **Disk Space**: Minimum 5GB
- **RAM**: Minimum 2GB

Local Development Setup

Step 1: Clone and Install Dependencies

```
# Navigate to project directory
cd shopping_website

# Install dependencies
pnpm install
```

Step 2: Set Up Environment Variables

Create a `.env.local` file in the root directory:

```
 # Database
DATABASE_URL=mysql://user:password@localhost:3306/nomas_db

# Authentication
JWT_SECRET=your-secret-key-here-min-32-characters

# OAuth (if using Manus OAuth)
VITE_APP_ID=your-app-id
OAUTH_SERVER_URL=https://api.manus.im
VITE_OAUTH_PORTAL_URL=https://portal.manus.im

# API Keys
BUILT_IN_FORGE_API_KEY=your-api-key
BUILT_IN_FORGE_API_URL=https://api.manus.im

# Frontend
VITE_APP_TITLE=NOMAS
VITE_APP_LOGO=/logo.png
```

Step 3: Create Database

```
# Create MySQL database
mysql -u root -p

# In MySQL console:
CREATE DATABASE nomas_db CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE USER 'nomas_user'@'localhost' IDENTIFIED BY 'secure_password';
GRANT ALL PRIVILEGES ON nomas_db.* TO 'nomas_user'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

Step 4: Run Database Migrations

```
# Push schema to database
pnpm db:push
```

Step 5: Start Development Server

```
# Start both frontend and backend
pnpm dev

# Frontend will be available at: http://localhost:5173
# Backend API at: http://localhost:3000
```

Database Configuration

Database Schema

The application includes the following tables:

Table	Purpose
`users`	User accounts and authentication
`products`	Product catalog
`cartItems`	Shopping cart items
`orders`	Customer orders
`orderItems`	Individual items in orders

Backup Database

```
# Backup database
mysqldump -u nomas_user -p nomas_db > nomas_backup.sql

# Restore database
mysql -u nomas_user -p nomas_db < nomas_backup.sql
```

Reset Database

```
# Drop and recreate database (WARNING: This deletes all data)
mysql -u root -p

# In MySQL console:
DROP DATABASE nomas_db;
CREATE DATABASE nomas_db CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
GRANT ALL PRIVILEGES ON nomas_db.* TO 'nomas_user'@'localhost';
FLUSH PRIVILEGES;
EXIT;

# Then run migrations
pnpm db:push
```

Environment Variables

Required Variables

Variable	Description	Example
`DATABASE_URL`	MySQL connection string	`mysql://user:pass@host:3306/db`
`JWT_SECRET`	Session encryption key (min 32 chars)	`your-secret-key-here`
`VITE_APP_TITLE`	Website title	`NOMAS`

Optional Variables

Variable	Description	Default
`VITE_APP_LOGO`	Logo path	`/logo.png`
`NODE_ENV`	Environment	`development`
`PORT`	Server port	`3000`

Deployment Guide

Option 1: Deploy on Shared Hosting (cPanel)

Prerequisites

- cPanel access
- SSH access
- Node.js support enabled

Steps

1. **Upload Files via FTP/SFTP** “`

- Connect to your hosting via SFTP
- Upload all files to public_html or a subdirectory
- Ensure node_modules is not uploaded (use .gitignore)

” `

2. **Install Dependencies**

```
ssh user@host
cd public_html
pnpm install --production
```

3. **Configure Environment**

```
# Create .env file with production variables
nano .env
```

4. **Build Frontend**

```
pnpm build
```

5. **Set Up Reverse Proxy (via cPanel)**

- Go to cPanel → Addon Domains or Subdomains
- Create domain/subdomain
- Configure reverse proxy to Node.js port (3000)

6. **Start Application**

```
# Using PM2 (recommended)
pnpm install -g pm2
pm2 start server/index.ts --name "nomas"
pm2 startup
pm2 save
```

Option 2: Deploy on VPS (Ubuntu 20.04+)

Prerequisites

- Root or sudo access
- Domain name
- SSL certificate (Let’ s Encrypt)

Steps

1. **Update System**

```
sudo apt update && sudo apt upgrade -y
```

2. **Install Node.js**

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install -y nodejs
npm install -g pnpm
```

3. **Install MySQL**

```
sudo apt install -y mysql-server
sudo mysql_secure_installation
```

4. **Clone Repository**

```
cd /var/www
git clone <your-repo-url> nomas
cd nomas
pnpm install
```

5. **Configure Environment**

```
sudo nano .env
# Add all environment variables
```

6. **Run Database Migrations**

```
pnpm db:push
```

7. **Build Application**

```
pnpm build
```

8. **Install Nginx**

```
sudo apt install -y nginx
```

9. **Configure Nginx**

```
sudo nano /etc/nginx/sites-available/nomas
```

Add this configuration:

```
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

```
sudo ln -s /etc/nginx/sites-available/nomas /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

1. **Set Up SSL (Let’s Encrypt)**

```
sudo apt install -y certbot python3-certbot-nginx
sudo certbot --nginx -d yourdomain.com -d www.yourdomain.com
```

2. **Start Application with PM2**

```
sudo npm install -g pm2
cd /var/www/nomas
pm2 start server/index.ts --name "nomas"
pm2 startup
pm2 save
```

3. **Enable Auto-restart on Reboot**

```
sudo systemctl enable pm2-root
```

Option 3: Deploy on Heroku

Steps

1. **Install Heroku CLI**

```
curl https://cli-assets.heroku.com/install.sh | sh
```

2. **Login to Heroku**

```
heroku login
```

3. **Create Heroku App**

```
heroku create nomas-app
```

4. **Add MySQL Database**

```
heroku addons:create cleardb:ignite
```

5. **Set Environment Variables**

```
heroku config:set JWT_SECRET=your-secret-key
heroku config:set VITE_APP_TITLE=NOMAS
```

6. **Deploy**

```
git push heroku main
```

Security Recommendations

1. Database Security

```
-- Create limited user for application
CREATE USER 'nomas_app'@'localhost' IDENTIFIED BY 'strong_password_here';
GRANT SELECT, INSERT, UPDATE, DELETE ON nomas_db.* TO
'nomas_app'@'localhost';
FLUSH PRIVILEGES;

-- Enable SSL for database connections
-- Update DATABASE_URL to use SSL
```

2. Environment Variables

- **Never commit `.env` files to Git**
- Use `.gitignore` to exclude sensitive files
- Store secrets in environment variables
- Rotate `JWT_SECRET` periodically

3. Input Validation

The application includes:

- Frontend validation using Zod
- Backend validation on all tRPC procedures
- SQL injection prevention via prepared statements (Drizzle ORM)
- CSRF protection via session cookies

4. HTTPS/SSL

- Always use HTTPS in production
- Obtain free SSL certificate from Let’ s Encrypt
- Redirect HTTP to HTTPS

- Set `Secure` flag on cookies

5. Password Security

- Passwords are hashed using bcrypt
- Minimum 8 characters recommended
- Enforce strong password policies

6. Rate Limiting

Implement rate limiting for:

- Login attempts (max 5 per 15 minutes)
- API endpoints (max 100 requests per minute)
- File uploads (max 5MB per file)

7. CORS Configuration

```
// In server configuration
const corsOptions = {
  origin: process.env.FRONTEND_URL,
  credentials: true,
  methods: ['GET', 'POST', 'PUT', 'DELETE'],
  allowedHeaders: ['Content-Type', 'Authorization'],
};
```

8. Security Headers

```
 # In Nginx configuration
add_header X-Frame-Options "SAMEORIGIN" always;
add_header X-Content-Type-Options "nosniff" always;
add_header X-XSS-Protection "1; mode=block" always;
add_header Referrer-Policy "no-referrer-when-downgrade" always;
add_header Content-Security-Policy "default-src 'self' https:; script-src
'self' 'unsafe-inline' https:;" always;
```

9. Regular Backups

```
# Automated daily backup script
#!/bin/bash
BACKUP_DIR="/backups/nomas"
DATE=$(date +%Y%m%d_%H%M%S)
mysqldump -u nomas_user -p$DB_PASSWORD nomas_db >
$BACKUP_DIR/nomas_$DATE.sql
gzip $BACKUP_DIR/nomas_$DATE.sql
```

10. Monitoring & Logging

- Monitor server CPU, memory, and disk usage
- Log all authentication attempts
- Log all database queries (in development)
- Set up alerts for errors and anomalies

Admin Setup

Creating an Admin User

Method 1: Direct Database Update

```
-- Update user role to admin
UPDATE users SET role = 'admin' WHERE id = 1;
```

Method 2: Via Application

1. Create a regular user account
2. SSH into server
3. Run database query above
4. User will have admin access on next login

Admin Dashboard Access

1. Log in with admin account
2. Navigate to `/admin` (to be implemented)
3. Access features:
    - Product management (add, edit, delete)
    - Order management
    - User management
    - Analytics

Default Admin Credentials

- **Email**: Set via first admin user
- **Password**: Set during account creation
- **Role**: `admin`

Troubleshooting

Common Issues

1. Database Connection Error

Error: `Error: connect ECONNREFUSED 127.0.0.1:3306`

Solution:

```
# Check MySQL is running
sudo systemctl status mysql

# Start MySQL if stopped
sudo systemctl start mysql

# Verify DATABASE_URL in .env
# Format: mysql://user:password@host:port/database
```

2. Port Already in Use

Error: `Error: listen EADDRINUSE: address already in use :::3000`

Solution:

```
# Find process using port 3000
lsof -i :3000

# Kill the process
kill -9 <PID>

# Or use different port
PORT=3001 pnpm dev
```

3. Authentication Fails

Error: `Authentication failed`

Solution:

- Verify JWT_SECRET is set correctly
- Check session cookie is being saved
- Clear browser cookies and try again
- Check OAuth configuration if using external auth

4. Build Fails

Error: `TypeScript compilation error`

Solution:

```
# Clear node_modules and reinstall
rm -rf node_modules pnpm-lock.yaml
pnpm install

# Run type check
pnpm type-check
```

5. Slow Performance

Solutions:

- Enable database query caching
- Implement Redis for session storage
- Optimize images and assets
- Enable gzip compression in Nginx
- Use CDN for static files

Maintenance

Regular Tasks

Task	Frequency	Command
Database backup	Daily	`mysqldump ...`
Update dependencies	Monthly	`pnpm update`
Security patches	As needed	`pnpm audit fix`
Log rotation	Weekly	Configure logrotate
Database optimization	Monthly	`OPTIMIZE TABLE ...`

Monitoring

```
# Monitor server resources
top

# Monitor disk usage
df -h

# Monitor database
mysql -u root -p -e "SHOW PROCESSLIST;"
```

Support & Resources

- **Documentation**: See README.md
- **Issues**: Check GitHub issues
- **Database**: MySQL documentation
- **Node.js**: nodejs.org
- **React**: react.dev
- **Tailwind CSS**: tailwindcss.com

License

Last Updated: November 2025 **Version**: 1.0.0