Step 3: Staging Testing
Test all **core flows**:
User registration and login.
Product search, cart, and checkout.
Verify **payment gateway integration** using sandbox credentials.
Step 4: Load and Performance Testing
Use **Apache JMeter** or **k6** for load testing.
Identify bottlenecks and optimize API performance.

3. Production Deployment
Step 1: Final Deployment
Push your changes to the **main branch** for deployment if using CI/CD.
Deploy to a reliable provider like:
Vercel for Next.js (zero-config).
AWS or DigitalOcean for more control.
Step 2: Domain Configuration
Point your **custom domain** to your deployment.
Use **DNS settings** for domain binding.
Step 3: Monitor and Scale
Set up **monitoring tools** like **Sentry** or **Datadog** for error tracking.
Enable **auto-scaling** if necessary.

Step 4: Security Enhancements
Ensure **HTTPS** is enforced.
Set up **CORS** to restrict
origins.javascriptCopy codeconst cors =
require('cors');
app.use(cors({ origin:
'https://yourdomain.com' }));

2. Staging Environment
Step 1: Set Up Staging Server
Use **Vercel**, **AWS**, or **Heroku** for staging.
Connect your repository for automatic deployments.
Step 2: Database for Staging
Use a **separate database** from production to avoid data corruption.
Example connection for **MongoDB Atlas** in .env.staging:

STAGING_DATABASE_URL=mongodb+srv://
staging-user:password@cluster.mongodb.net
/staging-db

Step 3: Staging Testing
Test all **core flows**:
User registration and login.
Product search, cart, and checkout.
Verify **payment gateway integration** using sandbox credentials.

issues before deployment.

Step 3: Code Quality Checks
Run **linting** to ensure clean code.bashCopy code npm run lint

Use **Prettier** or **ESLint** for consistent formatting.

Step 4: Security Enhancements
Ensure **HTTPS** is enforced.
Set up **CORS** to restrict origins.javascriptCopy codeconst cors = require('cors');
app.use(cors({ origin: 'https://yourdomain.com' }));

2. Staging Environment
Step 1: Set Up Staging Server
Use **Vercel**, **AWS**, or **Heroku** for staging. Connect your repository for automatic deployments.
Step 2: Database for Staging
Use a **separate database** from production to avoid data corruption.
Example connection for **MongoDB Atlas** in .env.staging:

STAGING_DATABASE_URL=mongodb+srv://

Hackathon Day06 Deployment Prepation And Staging For My Market Place Builder

1. Deployment Preparation

Step 1: Environment Configuration

Use **environment variables** for sensitive data.
Create a .env file for local use.
Use process.env for access in Node.js and Next.js.
Example .env file:plaintextCopy codeDATABASE_URL=mongodb://username:password@host:port/dbname
STRIPE_SECRET_KEY=your_secret_key
JWT_SECRET=your_jwt_secret

Step 2: Build Optimization

Run the Next.js build command to optimize static files.bashCopy codenpm run build

Check for warnings and errors to resolve any issues before deployment.

Step 3: Code Quality Checks

Run **linting** to ensure clean code.bashCopy code npm run lint

Use **Prettier** or **ESLint** for consistent formatting.