Deployment Guide

Complete guide for deploying the Homework Assignment System to production.

🎯 Pre-Deployment Checklist

✅ Code Readiness

- [ ] All tests passing
- [ ] Build successful (`yarn build`)
- [ ] Environment variables configured
- [ ] Database schema up to date
- [ ] Security review completed

✅ Infrastructure Requirements

- [ ] PostgreSQL database (14+)
- [ ] Node.js runtime (18+)
- [ ] SSL certificate for HTTPS
- [ ] Domain name configured
- [ ] Backup strategy in place

🚀 Deployment Options

Option 1: Vercel (Recommended)

Quick Deploy:

```
# Install Vercel CLI
npm i -g vercel

# Deploy from project root
vercel

# Follow prompts to configure
```

Environment Variables:

```
# Add to Vercel project settings
DATABASE_URL="postgresql://user:pass@host:5432/db"
SESSION_SECRET="your-production-secret-key"
```

Database Setup:

```
# Deploy schema to production database
npx prisma db push

# Seed production data (optional)
npx prisma db seed
```

Option 2: Docker Deployment

Create Dockerfile:

```
FROM node:18-alpine

WORKDIR /app
COPY package*.json ./
COPY yarn.lock ./
RUN yarn install --frozen-lockfile

COPY . .
RUN npx prisma generate
RUN yarn build

EXPOSE 3000
CMD ["yarn", "start"]
```

Docker Compose:

```
version: '3.8'
services:
  app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - DATABASE_URL=postgresql://user:pass@db:5432/homework
      - SESSION_SECRET=production-secret
    depends_on:
      - db

  db:
    image: postgres:14
    environment:
      - POSTGRES_DB=homework
      - POSTGRES_USER=user
      - POSTGRES_PASSWORD=pass
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  postgres_data:
```

Deploy:

```
docker-compose up -d
```

Option 3: VPS/Cloud Server

Server Setup (Ubuntu 22.04):

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install Node.js 18
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs

# Install Yarn
corepack enable
corepack prepare yarn@stable --activate

# Install PostgreSQL
sudo apt install postgresql postgresql-contrib
sudo systemctl start postgresql
sudo systemctl enable postgresql
```

Application Setup:

```
# Clone repository
git clone <your-repo-url> homework-system
cd homework-system/app

# Install dependencies
yarn install --production

# Set up environment
cp .env.example .env
nano .env  # Configure database URL and secrets

# Database setup
npx prisma db push
npx prisma db seed

# Build application
yarn build
```

Process Management (PM2):

```
# Install PM2
npm install -g pm2

# Create ecosystem file
cat > ecosystem.config.js << EOF
module.exports = {
  apps: [{
    name: 'homework-system',
    script: 'yarn',
    args: 'start',
    instances: 'max',
    exec_mode: 'cluster',
    env: {
      NODE_ENV: 'production',
      PORT: 3000
    }
  }]
};
EOF

# Start application
pm2 start ecosystem.config.js
pm2 startup
pm2 save
```

Nginx Reverse Proxy:

```
server {
    listen 80;
    server_name your-domain.com;

    location / {
        return 301 https://$server_name$request_uri;
    }
}

server {
    listen 443 ssl http2;
    server_name your-domain.com;

    ssl_certificate /path/to/certificate.crt;
    ssl_certificate_key /path/to/private.key;

    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_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
    }
}
```

🔒 Security Configuration

Environment Variables

```
# Production .env
DATABASE_URL="postgresql://user:secure_password@host:5432/homework_prod"
SESSION_SECRET="super-secure-random-string-64-characters-minimum"
NODE_ENV="production"
```

Database Security

```
-- Create dedicated database user
CREATE USER homework_app WITH PASSWORD 'secure_password';
CREATE DATABASE homework_production OWNER homework_app;

-- Grant minimal required permissions
GRANT CONNECT ON DATABASE homework_production TO homework_app;
GRANT USAGE ON SCHEMA public TO homework_app;
GRANT CREATE ON SCHEMA public TO homework_app;
```

SSL/HTTPS Setup

```
# Using Let's Encrypt (free SSL)
sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx -d your-domain.com
sudo systemctl enable certbot.timer
```

Firewall Configuration

```
# UFW firewall setup
sudo ufw enable
sudo ufw allow ssh
sudo ufw allow 80
sudo ufw allow 443
sudo ufw status
```

📊 Performance Optimization

Next.js Configuration

```
// next.config.js
const nextConfig = {
  output: 'standalone',
  poweredByHeader: false,
  compress: true,
  images: {
    unoptimized: true
  },
  experimental: {
    serverComponentsExternalPackages: ['@prisma/client']
  }
};
```

Database Optimization

```
-- Production indexes
CREATE INDEX CONCURRENTLY idx_assignments_active ON assignments(status) WHERE status =
'active';
CREATE INDEX CONCURRENTLY idx_student_work_recent ON student_work(last_saved_at DESC);
CREATE INDEX CONCURRENTLY idx_teachers_username_hash ON teachers USING hash(username);

-- Analyze statistics
ANALYZE;
```

Caching Strategy

```
# Redis for session storage (optional)
sudo apt install redis-server
sudo systemctl enable redis-server
```

🔍 Monitoring Setup

Application Monitoring

```
// Add to next.config.js for analytics
const nextConfig = {
  experimental: {
    instrumentationHook: true
  }
};
```

Database Monitoring

```
-- Monitor active connections
SELECT count(*) as active_connections FROM pg_stat_activity;

-- Monitor table sizes
SELECT schemaname, tablename, pg_size_pretty(pg_total_relation_size(tablename::regclass
)) as size
FROM pg_tables WHERE schemaname = 'public' ORDER BY pg_total_relation_size(tablename::r
egclass) DESC;
```

Log Management

```
# Rotate application logs
sudo nano /etc/logrotate.d/homework-system

/var/log/homework-system/*.log {
    daily
    missingok
    rotate 52
    compress
    delaycompress
    notifempty
    copytruncate
}
```

💾 Backup Strategy

Database Backups

```
#!/bin/bash
# daily-backup.sh
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_DIR="/backups/homework-system"
DB_NAME="homework_production"

mkdir -p $BACKUP_DIR

pg_dump -h localhost -U homework_app -W $DB_NAME | gzip > "$BACKUP_DIR/backup_$DATE.sql
.gz"

# Keep only last 30 days
find $BACKUP_DIR -name "backup_*.sql.gz" -mtime +30 -delete
```

Schedule with cron:

```
# Add to crontab
crontab -e
0 2 * * * /path/to/daily-backup.sh
```

Application Backups

```
# Backup application files
tar -czf app_backup_$(date +%Y%m%d).tar.gz /path/to/homework-system
```

🔄 CI/CD Pipeline

GitHub Actions Example

```yaml
# .github/workflows/deploy.yml
name: Deploy to Production

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'

      - name: Install dependencies
        run: yarn install --frozen-lockfile

      - name: Run tests
        run: yarn test

      - name: Build application
        run: yarn build

      - name: Deploy to Vercel
        uses: amondnet/vercel-action@v25
        with:
          vercel-token: ${{ secrets.VERCEL_TOKEN }}
          vercel-org-id: ${{ secrets.ORG_ID }}
          vercel-project-id: ${{ secrets.PROJECT_ID }}
          vercel-args: '--prod'
```

📋 Post-Deployment Tasks

Initial Setup

1. **Create admin account:**
   `bash`
   `# Access production site and register first teacher`

2. **Test all features:**
   - Teacher registration/login
   - Assignment creation
   - Student access
   - Copy protection
   - File downloads

3. **Configure monitoring:**
   - Set up uptime monitoring

- Configure error alerting
- Monitor database performance

Security Hardening

```
# Disable root login
sudo nano /etc/ssh/sshd_config
# Set: PermitRootLogin no

# Update system packages
sudo apt update && sudo apt upgrade -y

# Configure automatic security updates
sudo dpkg-reconfigure -plow unattended-upgrades
```

Performance Testing

```
# Load testing with artillery
npm install -g artillery
artillery quick --count 100 --num 10 https://your-domain.com
```

🆘 Troubleshooting

Common Issues

Build Failures:

```
# Clear cache and rebuild
yarn clean
rm -rf .next node_modules
yarn install
yarn build
```

Database Connection Issues:

```
# Test database connection
npx prisma db push --preview-feature
```

Memory Issues:

```
# Increase Node.js memory limit
NODE_OPTIONS="--max-old-space-size=4096" yarn build
```

Health Checks

```
# Application health
curl -f http://localhost:3000/api/health || exit 1

# Database health
pg_isready -h localhost -p 5432
```

📞 Production Support

For production issues:
1. Check application logs
2. Monitor database performance
3. Verify SSL certificate status
4. Test backup restoration
5. Review security logs

Remember: Always test deployment procedures in a staging environment first!