🚀 Phase 2: Production Deployment Guide

📋 Deployment Overview

Deployment Strategy

- **Platform**: Vercel (recommended) + PostgreSQL
- **Database**: Supabase PostgreSQL or Railway
- **Domain**: Custom domain with SSL
- **Monitoring**: Built-in health checks
- **Backup**: Automated database backups

🔧 Step 1: Environment Setup

1.1 Production Environment Variables

Create `.env.production`:

```
# Database (Production)
DATABASE_URL="postgresql://username:password@host:5432/
sss_surplus_prod"

# NextAuth (Production)
NEXTAUTH_URL="https://yourdomain.com"
NEXTAUTH_SECRET="your-super-secure-production-secret-key-here"

# Super Admin (Production)
SUPER_ADMIN_MODE="true"
SUPER_ADMIN_EMAILS="sanchai5651@gmail.com"
SUPER_ADMIN_PASSWORD="Safety17"
SUPER_ADMIN_NAME="System Administrator"
NEXT_PUBLIC_SUPER_ADMIN_ENABLED="true"

# OAuth Providers (Production)
GOOGLE_CLIENT_ID="your-production-google-client-id"
GOOGLE_CLIENT_SECRET="your-production-google-client-secret"
FACEBOOK_CLIENT_ID="your-production-facebook-client-id"
FACEBOOK_CLIENT_SECRET="your-production-facebook-client-secret"
LINE_CLIENT_ID="your-production-line-client-id"
```

```
LINE_CLIENT_SECRET="your-production-line-client-secret"

# Email Service (Production)
SMTP_HOST="smtp.gmail.com"
SMTP_PORT="587"
SMTP_USER="noreply@yourdomain.com"
SMTP_PASS="your-production-app-password"
SMTP_FROM="SSS Surplus <noreply@yourdomain.com>"

# Security
NODE_ENV="production"
NEXT_PUBLIC_APP_URL="https://yourdomain.com"

# Monitoring
SENTRY_DSN="your-sentry-dsn-for-error-tracking"
```

1.2 OAuth Provider Configuration

Google OAuth Setup:

1. Go to Google Cloud Console
2. Create new project or select existing
3. Enable Google+ API
4. Create OAuth 2.0 credentials
5. Add authorized redirect URIs:
6. `https://yourdomain.com/api/auth/callback/google`

Facebook OAuth Setup:

1. Go to Facebook Developers
2. Create new app
3. Add Facebook Login product
4. Configure Valid OAuth Redirect URIs:
5. `https://yourdomain.com/api/auth/callback/facebook`

Line OAuth Setup:

1. Go to Line Developers
2. Create new channel (Line Login)
3. Configure Callback URL:
4. `https://yourdomain.com/api/auth/callback/line`

🗄️ Step 2: Database Setup

2.1 PostgreSQL Database Creation

Option A: Supabase (Recommended)

```
# 1. Create Supabase project
# 2. Get connection string
# 3. Update DATABASE_URL in environment
```

Option B: Railway

```
# 1. Create Railway account
# 2. Create PostgreSQL service
# 3. Get connection string
# 4. Update DATABASE_URL in environment
```

Option C: Self-hosted

```
# Install PostgreSQL
sudo apt update
sudo apt install postgresql postgresql-contrib

# Create database
sudo -u postgres createdb sss_surplus_prod

# Create user
sudo -u postgres createuser --interactive sss_user
```

2.2 Run Production Migrations

```
#!/bin/bash
# production-migration.sh

echo "🗄️ Running Production Database Migrations..."

# Set production database URL
export DATABASE_URL="your-production-database-url"

# Run migrations in order
psql $DATABASE_URL -f 001_create_users_table.sql
psql $DATABASE_URL -f 002_create_accounts_table.sql
psql $DATABASE_URL -f 003_create_sessions_table.sql
psql $DATABASE_URL -f 004_create_verification_tokens_table.sql
psql $DATABASE_URL -f 005_create_role_assignments_table.sql
psql $DATABASE_URL -f 006_create_admin_actions_table.sql
psql $DATABASE_URL -f 007_create_login_logs_table.sql
psql $DATABASE_URL -f 008_create_notifications_table.sql
psql $DATABASE_URL -f 009_insert_super_admin.sql

echo "✅ Production migrations completed!"

# Verify Super Admin
psql $DATABASE_URL -c "SELECT email, role FROM users WHERE role
= 'super_admin';"
```

🚀 Step 3: Vercel Deployment

3.1 Vercel Configuration

Create `vercel.json`:

```json
{
  "version": 2,
  "builds": [
    {
      "src": "package.json",
      "use": "@vercel/next"
    }
  ],
  "routes": [
    {
      "src": "/api/(.*)",
      "dest": "/api/$1"
    },
    {
      "src": "/(.*)",
      "dest": "/$1"
    }
  ],
  "env": {
    "NEXT_PUBLIC_SUPER_ADMIN_ENABLED": "true",
    "NEXT_PUBLIC_APP_URL": "https://yourdomain.com"
  },
  "functions": {
    "src/pages/api/**/*.js": {
      "maxDuration": 30
    }
  }
}
```

3.2 Deployment Script

Create `deploy.sh`:

```bash
#!/bin/bash
# deploy.sh - Production Deployment Script

echo "🚀 Starting Production Deployment..."

# 1. Install Vercel CLI
npm install -g vercel

# 2. Login to Vercel
vercel login

# 3. Link project
vercel link

# 4. Set environment variables
echo "Setting environment variables..."
vercel env add DATABASE_URL production
vercel env add NEXTAUTH_SECRET production
vercel env add GOOGLE_CLIENT_ID production
vercel env add GOOGLE_CLIENT_SECRET production
vercel env add FACEBOOK_CLIENT_ID production
vercel env add FACEBOOK_CLIENT_SECRET production
vercel env add LINE_CLIENT_ID production
vercel env add LINE_CLIENT_SECRET production
vercel env add SMTP_HOST production
vercel env add SMTP_USER production
vercel env add SMTP_PASS production

# 5. Deploy to production
vercel --prod

echo "✅ Deployment completed!"
echo "🌐 Your app is live at: https://yourdomain.com"
```

3.3 Custom Domain Setup

```bash
# Add custom domain
vercel domains add yourdomain.com

# Configure DNS
# Add CNAME record: www -> cname.vercel-dns.com
# Add A record: @ -> 76.76.19.61
```

🔒 Step 4: SSL & Security

4.1 SSL Certificate

- Vercel automatically provides SSL certificates
- Custom domains get automatic HTTPS
- Certificate renewal is automatic

4.2 Security Headers

Create `next.config.js`:

```
/** @type {import('next').NextConfig} */
const nextConfig = {
  async headers() {
    return [
      {
        source: '/(.*)',
        headers: [
          {
            key: 'X-Frame-Options',
            value: 'DENY',
          },
          {
            key: 'X-Content-Type-Options',
            value: 'nosniff',
          },
          {
            key: 'Referrer-Policy',
            value: 'origin-when-cross-origin',
          },
          {
            key: 'Strict-Transport-Security',
            value: 'max-age=31536000; includeSubDomains',
          },
        ],
      },
    ];
  },
  env: {
    SUPER_ADMIN_ENABLED:
process.env.NEXT_PUBLIC_SUPER_ADMIN_ENABLED,
  },
};

module.exports = nextConfig;
```

📊 Step 5: Monitoring & Health Checks

5.1 Health Check Endpoint

Create `pages/api/health.js`:

```
// pages/api/health.js
import { PrismaClient } from '@prisma/client';

const prisma = new PrismaClient();

export default async function handler(req, res) {
  if (req.method !== 'GET') {
    return res.status(405).json({ error: 'Method not
allowed' });
  }

  try {
    // Check database connection
    await prisma.$queryRaw`SELECT 1`;

    // Check environment variables
    const requiredEnvs = [
      'DATABASE_URL',
      'NEXTAUTH_SECRET',
      'SUPER_ADMIN_EMAILS',
      'SUPER_ADMIN_PASSWORD'
    ];

    const missingEnvs = requiredEnvs.filter(env => !
process.env[env]);

    if (missingEnvs.length > 0) {
      return res.status(500).json({
        status: 'error',
        message: 'Missing environment variables',
        missing: missingEnvs
      });
    }

    // Check Super Admin exists
    const superAdmin = await prisma.user.findFirst({
      where: { role: 'super_admin' }
    });

    if (!superAdmin) {
      return res.status(500).json({
        status: 'error',
        message: 'Super Admin not found in database'
      });
    }

    res.status(200).json({
      status: 'healthy',
      timestamp: new Date().toISOString(),
      database: 'connected',
      superAdmin: 'exists',
      environment: process.env.NODE_ENV
    });

  } catch (error) {
    console.error('Health check failed:', error);
    res.status(500).json({
      status: 'error',
      message: 'Health check failed',
      error: error.message
    });
  } finally {
    await prisma.$disconnect();
  }
}
```

5.2 Monitoring Script

Create `monitor.js`:

```
// monitor.js - Production Monitoring
const https = require('https');

const HEALTH_CHECK_URL = 'https://yourdomain.com/api/health';
const CHECK_INTERVAL = 5 * 60 * 1000; // 5 minutes

function checkHealth() {
  const startTime = Date.now();

  https.get(HEALTH_CHECK_URL, (res) => {
    const responseTime = Date.now() - startTime;
    let data = '';

    res.on('data', (chunk) => {
      data += chunk;
    });

    res.on('end', () => {
      try {
        const result = JSON.parse(data);

        if (res.statusCode === 200 && result.status ===
'healthy') {
          console.log(`✅ Health check passed (${responseTime}
ms)`);
        } else {
          console.error(`❌ Health check failed:`, result);
          // Send alert (email, Slack, etc.)
        }
      } catch (error) {
        console.error(`❌ Health check parse error:`, error);
      }
    });
  }).on('error', (error) => {
    console.error(`❌ Health check request failed:`, error);
    // Send alert
  });
}

// Run health check every 5 minutes
setInterval(checkHealth, CHECK_INTERVAL);
checkHealth(); // Run immediately

console.log(`🔍 Monitoring started for ${HEALTH_CHECK_URL}`);
```

🔄 Step 6: Backup & Recovery

6.1 Database Backup Script

Create `backup.sh`:

```
#!/bin/bash
# backup.sh - Database Backup Script

BACKUP_DIR="/backups/sss-surplus"
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_FILE="$BACKUP_DIR/sss_surplus_backup_$DATE.sql"

# Create backup directory
mkdir -p $BACKUP_DIR

# Create database backup
pg_dump $DATABASE_URL > $BACKUP_FILE

# Compress backup
gzip $BACKUP_FILE

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

echo "✅ Backup completed: $BACKUP_FILE.gz"
```

6.2 Automated Backup (Cron)

```
# Add to crontab (crontab -e)
# Daily backup at 2 AM
0 2 * * * /path/to/backup.sh

# Weekly full backup on Sunday at 1 AM
0 1 * * 0 /path/to/full-backup.sh
```

🧪 Step 7: Post-Deployment Testing

7.1 Production Test Script

Create `production-test.js`:

```
// production-test.js
const https = require('https');

const PRODUCTION_URL = 'https://yourdomain.com';

const tests = [
  {
    name: 'Health Check',
    path: '/api/health',
    expectedStatus: 200
  },
  {
    name: 'Home Page',
    path: '/',
    expectedStatus: 200
  },
  {
    name: 'Auth Signin',
    path: '/auth/signin',
    expectedStatus: 200
  },
  {
    name: 'Super Admin Login API',
    path: '/api/auth/super-admin-login',
    method: 'POST',
    expectedStatus: 405 // Method not allowed for GET
  }
];

async function runTest(test) {
  return new Promise((resolve) => {
    const options = {
      hostname: new URL(PRODUCTION_URL).hostname,
      path: test.path,
      method: test.method || 'GET'
    };

    const req = https.request(options, (res) => {
      const success = res.statusCode === test.expectedStatus;
      console.log(`${success ? '✅' : '❌'} ${test.name}: $
{res.statusCode}`);
      resolve(success);
    });

    req.on('error', (error) => {
      console.log(`❌ ${test.name}: ${error.message}`);
      resolve(false);
    });

    req.end();
  });
}

async function runAllTests() {
  console.log('🧪 Running Production Tests...\n');

  const results = await Promise.all(tests.map(runTest));
  const passed = results.filter(Boolean).length;

  console.log(`\n📊 Test Results: ${passed}/${tests.length}
passed`);

  if (passed === tests.length) {
    console.log('🎉 All tests passed! Production is ready.');
  } else {
    console.log('⚠️ Some tests failed. Please check the
issues.');
  }
}

runAllTests();
```

📋 Step 8: Go-Live Checklist

Pre-Launch Checklist

- [ ] Database migrations completed
- [ ] Environment variables configured
- [ ] OAuth providers configured and tested

- [ ] SSL certificate active
- [ ] Custom domain configured
- [ ] Health checks passing
- [ ] Backup system configured
- [ ] Monitoring active
- [ ] Super Admin login tested
- [ ] Social login tested
- [ ] Email notifications tested
- [ ] Performance testing completed
- [ ] Security testing completed

Launch Day Checklist

- [ ] Final deployment completed
- [ ] DNS propagation verified
- [ ] All tests passing
- [ ] Monitoring alerts configured
- [ ] Support team notified
- [ ] Documentation updated
- [ ] User communication sent

Post-Launch Checklist

- [ ] Monitor error rates
- [ ] Check performance metrics
- [ ] Verify user registrations
- [ ] Test critical user flows
- [ ] Monitor email delivery
- [ ] Check database performance
- [ ] Verify backup completion

🚨 Troubleshooting

Common Issues

1. Database Connection Issues

```
# Test database connection
psql $DATABASE_URL -c "SELECT version();"
```

```
# Check connection limits
psql $DATABASE_URL -c "SELECT count(*) FROM pg_stat_activity;"
```

2. OAuth Login Failures

- Verify redirect URIs match exactly
- Check client ID and secret
- Ensure OAuth apps are published/approved

3. Email Delivery Issues

```
// Test email configuration
const nodemailer = require('nodemailer');

const transporter = nodemailer.createTransporter({
  host: process.env.SMTP_HOST,
  port: process.env.SMTP_PORT,
  auth: {
    user: process.env.SMTP_USER,
    pass: process.env.SMTP_PASS
  }
});

transporter.verify((error, success) => {
  if (error) {
    console.log('❌ Email config error:', error);
  } else {
    console.log('✅ Email server ready');
  }
});
```

4. Performance Issues

- Enable database connection pooling
- Implement Redis caching
- Optimize database queries
- Use CDN for static assets

📊 Success Metrics

Key Performance Indicators

- **Uptime**: > 99.9%
- **Response Time**: < 500ms

- **Error Rate**: < 0.1%
- **User Registration Success**: > 95%
- **Email Delivery Rate**: > 98%

Monitoring Dashboards

- Application performance
- Database performance
- User authentication metrics
- Error tracking
- Business metrics

🎉 Deployment Complete

Once all steps are completed, your SSS Surplus Marketplace Role-based Authentication system will be live in production with:

✅ **Secure Authentication** - Super Admin + Social Login ✅ **Role Management** - Hierarchical permissions ✅ **Production Database** - Reliable and backed up ✅ **SSL Security** - HTTPS everywhere ✅ **Monitoring** - Health checks and alerts ✅ **Scalability** - Ready for growth

Production URL: https://yourdomain.com **Admin Access**: https://yourdomain.com/auth/signin **Health Check**: https://yourdomain.com/api/health

Your system is now ready to serve users! 🚀