DSP Quotation System - Installation Guide

A modern quotation management system built with Next.js, TypeScript, Prisma, and MySQL.

📋 Prerequisites

Before you begin, ensure you have the following installed on your system:

- **Node.js** (version 18 or higher)

```
node --version  # Should be 18.0.0 or higher
```

- **npm** (comes with Node.js) or **yarn** or **pnpm**
- **MySQL** (version 8.0 or higher)
- **Git** (for cloning the repository)

🚀 Installation Steps

1. Clone the Repository

```
git clone <repository-url>
cd dsp-quotation-system
```

2. Install Dependencies

```
npm install
# or
yarn install
# or
pnpm install
```

3. Database Setup

Create MySQL Database

1. Start your MySQL server
2. Create a new database for the application:

```
CREATE DATABASE dsp_quotation_system;
```

Set up Environment Variables

Create a `.env` file in the root directory of the project:

```
touch .env
```

Add the following environment variables to your `.env` file:

```
# Database Configuration
DATABASE_URL="mysql://username:password@localhost:3306/dsp_quotation_syste
m"

MYSQL_HOST=localhost
   MYSQL_USER=root
   MYSQL_PASSWORD=root
   MYSQL_DATABASE=quotation_system
```

Important: Replace the placeholder values with your actual configuration:

- Update `username` and `password` in the DATABASE_URL with your MySQL credentials
- Generate a secure secret for `NEXTAUTH_SECRET`
- Configure Google OAuth credentials if using Google authentication
- Set up Resend API key if using email functionality

4. Database Migration

Run Prisma migrations to set up your database schema:

```
# Generate Prisma client
npx prisma generate

# Run database migrations
npx prisma migrate dev

# (Optional) Seed the database if seed file exists
npx prisma db seed
```

5. Start the Development Server

```
npm run dev
# or
yarn dev
# or
pnpm dev
```

The application will be available at http://localhost:3000

📁 Project Structure

```
dsp-quotation-system/
├── src/
│   ├── app/                    # Next.js App Router pages
│   ├── components/             # React components
│   ├── lib/                    # Utility functions and configurations
│   ├── hooks/                  # Custom React hooks
│   ├── types/                  # TypeScript type definitions
│   └── middleware.ts           # Next.js middleware for authentication
├── prisma/
│   ├── schema.prisma           # Database schema
│   └── migrations/             # Database migrations
├── public/                     # Static assets
├── generated/                  # Generated Prisma client
├── package.json                # Project dependencies and scripts
└── .env                        # Environment variables (create this)
```

🔧 Available Scripts

Command	Description
`npm run dev`	Start development server with Turbopack
`npm run build`	Build the application for production
`npm run start`	Start the production server
`npm run lint`	Run ESLint for code linting
`npx prisma studio`	Open Prisma Studio for database management
`npx prisma migrate dev`	Run database migrations
`npx prisma generate`	Generate Prisma client

🗄️ Database Features

The application includes the following main database models:

- **Quotations** - Main quotation records with items and terms
- **Quotation Items** - Individual items within quotations
- **Terms** - Terms and conditions for quotations
- **Action History** - Audit trail for quotation changes
- **Quotation Requests** - Customer quotation requests
- **Request Action History** - Audit trail for request changes

🔐 Authentication & Authorization

The system includes:

- **Cookie-based authentication** with user sessions
- **Role-based access control** (internal/external users)
- **Company-based data isolation** for external users
- **Route protection** via middleware

User Types:

- **Internal Users**: Full system access (various departments)
- **External Users**: Limited access to company-specific quotations

🎨 UI/UX Features

- **Modern, responsive design** with Tailwind CSS
- **Dark/Light theme support** with next-themes
- **Component library** using Radix UI primitives
- **Data tables** with sorting and filtering (@tanstack/react-table)
- **Form handling** with react-hook-form and Zod validation
- **Toast notifications** with Sonner
- **PDF generation** with @react-pdf/renderer
- **File uploads** with react-dropzone
- **Smooth animations** with Framer Motion

🚀 Production Deployment

Build for Production

```
npm run build
npm run start
```

Environment Variables for Production

Ensure you update your `.env` file for production with:

- Production database URL
- Secure NEXTAUTH_SECRET
- Production domain for NEXTAUTH_URL
- Production API keys for external services

Database in Production

1. Set up your production MySQL database
2. Update the `DATABASE_URL` in your production environment
3. Run migrations:

```
npx prisma migrate deploy
npx prisma generate
```

🔍 Troubleshooting

Common Issues

1. **Database Connection Error**
   - Verify MySQL is running
   - Check DATABASE_URL format and credentials
   - Ensure database exists
2. **Prisma Client Issues**
   - Run `npx prisma generate` to regenerate client
   - Clear `node_modules` and reinstall dependencies
3. **Port Already in Use**
   - Change the port: `npm run dev -- -p 3001`
   - Or kill the process using port 3000
4. **Environment Variables Not Loading**
   - Ensure `.env` file is in the root directory
   - Restart the development server after changes
   - Check for syntax errors in `.env` file

Logs and Debugging

- Check browser console for frontend errors
- Use `npx prisma studio` to inspect database
- Enable verbose logging by setting `NODE_ENV=development`

🔄 Updates

To update the project:

```
# Pull latest changes
git pull origin main

# Install new dependencies
npm install

# Run any new migrations
npx prisma migrate dev

# Regenerate Prisma client
npx prisma generate
```

Note: This is a development installation guide. For production deployment, additional considerations for security, performance, and scalability should be implemented.