QR Code Product Catalogue Manager

A modern Next.js application that allows you to create digital product catalogues with QR codes for easy sharing. Features full CRUD operations, image support, and real-time updates with Socket.IO.

✨ Features

🗂️ Catalogue Management

- **Create Catalogues**: Build custom product catalogues with titles and descriptions
- **Read Catalogues**: View all catalogues with their products and statistics
- **Update Catalogues**: Edit catalogue titles and descriptions
- **Delete Catalogues**: Remove catalogues (also deletes associated products)

📦 Product Management

- **Create Products**: Add products with names, descriptions, images, and download links
- **Read Products**: View products within their catalogues with image previews
- **Update Products**: Edit product names, descriptions, images, and download URLs
- **Delete Products**: Remove individual products from catalogues
- **Image Support**: Upload and preview product images with full-screen modal
- **Image Preview**: Click on any product image to view it in full-screen

🔗 QR Code Generation

- **Generate QR Codes**: Automatically generate QR codes that link to your catalogues
- **Regenerate QR Codes**: Update QR codes when needed
- **Mobile-Friendly Display**: Beautiful responsive catalogue pages for mobile and desktop
- **Download Products**: Users can download product files directly from the catalogue

🎨 Modern UI/UX

- **Responsive Design**: Works perfectly on desktop, tablet, and mobile
- **Dark/Light Theme**: Beautiful UI with shadcn/ui components
- **Real-time Updates**: Socket.IO integration for live updates
- **Toast Notifications**: User-friendly feedback for all actions
- **Image Previews**: Full-screen image viewing with error handling

How It Works

Catalogue Management

1. **Create a Catalogue**:
    - Click "Create New Catalogue" on the home page
    - Enter a title and description for your catalogue
    - Click "Create Catalogue"

2. **Read/View Catalogues**:
    - All catalogues are displayed on the home page
    - Each catalogue shows its title, description, and product count
    - Click edit/delete buttons to manage catalogues
3. **Update a Catalogue**:
    - Click the edit icon (pencil) on any catalogue card
    - Modify the title and/or description
    - Click "Update Catalogue"
4. **Delete a Catalogue**:
    - Click the delete icon (trash) on any catalogue card
    - Confirm deletion in the popup dialog
    - Note: This will also delete all products in the catalogue

Product Management

1. **Create a Product**:
    - Click "Add Product" on any catalogue card
    - Enter product details (name, description)
    - Add an image URL for product visualization
    - Optionally add a download URL for the product file
    - Click "Add Product"
2. **Read/View Products**:
    - Products are listed within their catalogue cards with thumbnail images
    - Click on product images to view them in full-screen
    - Hover over products to see edit/delete options
    - Products display with their names, images, and download availability
3. **Update a Product**:
    - Hover over a product in the catalogue list
    - Click the edit icon (pencil) that appears
    - Modify product details as needed
    - Click "Update Product"
4. **Delete a Product**:
    - Hover over a product in the catalogue list
    - Click the delete icon (trash) that appears
    - Confirm deletion in the popup dialog

QR Code Generation

1. **Generate QR Code**:

    - Click "Generate QR" on any catalogue card
    - A QR code will be generated that links to your catalogue

2. **Share and Download**:
    - Share the QR code with customers
    - When scanned, users will see your catalogue on their mobile device
    - They can browse products and download files if available

3. **Automatic URL Detection**:
    - The system automatically detects the current domain when generating QR codes
    - This ensures QR codes work correctly in both development and production
    - No manual configuration required for most deployment scenarios

Technology Stack

- **Frontend**: Next.js 15 with TypeScript
- **Styling**: Tailwind CSS with shadcn/ui components
- **Database**: SQLite (local) / PostgreSQL (production) with Prisma ORM
- **Real-time**: Socket.IO for live updates
- **QR Code Generation**: qrcode library
- **Icons**: Lucide React
- **Package Manager**: pnpm
- **Deployment**: Vercel with Neon PostgreSQL

API Endpoints

Catalogue Endpoints

- `GET /api/catalogues` - Get all catalogues
- `POST /api/catalogues` - Create a new catalogue
- `GET /api/catalogues/[id]` - Get a specific catalogue
- `PUT /api/catalogues/[id]` - Update a catalogue
- `DELETE /api/catalogues/[id]` - Delete a catalogue

Product Endpoints

- `POST /api/catalogues/[id]/products` - Add a product to a catalogue
- `PUT /api/catalogues/[id]/products/[productId]` - Update a product
- `DELETE /api/catalogues/[id]/products/[productId]` - Delete a product

QR Code Endpoints

- `POST /api/catalogues/[id]/qrcode` - Generate QR code for a catalogue

Utility Endpoints

- `POST /api/migrate` - Test database connection and run migrations

Database Schema

Catalogue

- `id` - Unique identifier
- `title` - Catalogue title
- `description` - Catalogue description
- `qrCodeUrl` - URL to the generated QR code image
- `createdAt` - Creation timestamp
- `updatedAt` - Last update timestamp

Product

- `id` - Unique identifier
- `name` - Product name
- `description` - Product description
- `image` - Product image URL (optional)
- `fileUrl` - Download file URL (optional)
- `catalogueId` - Associated catalogue ID
- `createdAt` - Creation timestamp
- `updatedAt` - Last update timestamp

Getting Started

Prerequisites

- Node.js 18+
- pnpm (recommended) or npm
- Git

Local Development

1. **Clone the repository:**

```
git clone <your-repo-url>
cd QRCode
```

2. **Install dependencies:**

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

3. **Set up the database:**

```
pnpm run db:push
# or
npm run db:push
```

4. **Start the development server:**

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

5. **Open your browser:**
   - Main app: http://localhost:3003
   - Socket.IO: ws://localhost:3003/api/socketio

Environment Configuration

The application automatically detects the current domain for QR code generation. However, if you need to explicitly set the base URL (for example, in custom deployment scenarios), you can create a `.env.local` file:

```
# Copy the example environment file
cp .env.example .env.local

# Edit .env.local with your domain
NEXT_PUBLIC_BASE_URL=https://your-domain.com
```

Note: In most cases, you don't need to set this variable as the system will automatically detect the correct URL from the request headers.

🚀 Deployment

Deploy to Vercel (Recommended)

This application is optimized for Vercel deployment with Neon PostgreSQL.

Prerequisites

- Vercel account
- Neon account (for PostgreSQL database)
- GitHub repository (optional, for automatic deployments)

Step 1: Set Up Database

1. **Create a Neon account** at neon.tech

2. **Create a new project** in Neon
3. **Copy the connection string** from your Neon dashboard

Step 2: Deploy to Vercel

Option A: Deploy via Vercel CLI

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

# Login to Vercel
vercel login

# Deploy
vercel
```

Option B: Deploy via GitHub

1. Push your code to GitHub
2. Go to vercel.com
3. Click "New Project"
4. Import your GitHub repository
5. Configure build settings:
    - **Build Command**: `pnpm run build`
    - **Output Directory**: `.next`
    - **Install Command**: `pnpm install`

Step 3: Configure Environment Variables

In your Vercel project settings, add:

- **Key**: `DATABASE_URL`
- **Value**: Your Neon PostgreSQL connection string

  ```
  postgresql://username:password@host/database?sslmode=require
  ```

- **Key**: `NODE_ENV`
- **Value**: `production`

Step 4: Run Database Migration

After deployment, test your database connection:

```
# Visit your deployed app
https://your-app.vercel.app/api/migrate
```

Alternative Deployment Options

Deploy to Railway

1. Connect your GitHub repository to Railway
2. Add PostgreSQL database
3. Set environment variables
4. Deploy

Deploy to Netlify

1. Build the project: `pnpm run build`
2. Deploy the `.next` folder
3. Set up serverless functions for API routes

Environment Variables

Variable	Description	Required
`DATABASE_URL`	PostgreSQL connection string	Yes
`NODE_ENV`	Environment (production/development)	No
`NEXT_PUBLIC_BASE_URL`	Base URL for QR codes	No (auto-detected)

Usage Examples

Example Use Cases

1. **Digital Product Catalogue**: Create a catalogue for your digital products (e-books, software, templates) with download links and full CRUD management
2. **Restaurant Menu**: Create a menu catalogue with QR codes on tables for customers to view and download, with easy menu updates
3. **Real Estate Listings**: Create property catalogues with downloadable brochures, allowing agents to update listings and remove sold properties
4. **Event Materials**: Create catalogues for events with downloadable schedules and materials, with the ability to update content as needed
5. **Educational Resources**: Create learning material catalogues with downloadable resources, allowing instructors to update course materials

CRUD Operations Summary

Catalogue CRUD

- **Create**: New catalogue with title and description
- **Read**: View all catalogues or individual catalogue details
- **Update**: Modify existing catalogue information
- **Delete**: Remove catalogue and all associated products

Product CRUD

- **Create**: Add new product to existing catalogue
- **Read**: View products within their catalogue context
- **Update**: Modify existing product information
- **Delete**: Remove individual products from catalogue

🔧 Troubleshooting

Common Issues

Database Connection Issues

```
# Error: Database connection failed
# Solution: Check your DATABASE_URL environment variable
echo $DATABASE_URL
```

Build Failures

```
# Error: Prisma client not generated
# Solution: Run Prisma generate
pnpm run db:generate
```

QR Code Not Working

- Ensure your app is deployed and accessible
- Check that the base URL is correctly detected
- Verify the catalogue ID exists in the database

Image Upload Issues

- Ensure image URLs are publicly accessible
- Check that URLs use HTTPS in production
- Verify image format is supported (JPG, PNG, GIF)

Development Tips

1. **Hot Reload**: The app uses nodemon for automatic restarts

2. **Database Reset**: Use `pnpm run db:reset` to reset your local database
3. **Logs**: Check the terminal for Prisma query logs
4. **Socket.IO**: WebSocket connection is available at `/api/socketio`

File Structure

```
QRCode/
├── src/
│   ├── app/
│   │   ├── api/
│   │   │   ├── catalogues/
│   │   │   │   ├── [id]/
│   │   │   │   │   ├── products/
│   │   │   │   │   │   ├── [productId]/
│   │   │   │   │   │   │   └── route.ts
│   │   │   │   │   │   └── route.ts
│   │   │   │   │   ├── qrcode/
│   │   │   │   │   │   └── route.ts
│   │   │   │   │   └── route.ts
│   │   │   │   └── route.ts
│   │   │   └── migrate/
│   │   │       └── route.ts
│   │   ├── catalogue/
│   │   │   └── [id]/
│   │   │       └── page.tsx
│   │   ├── page.tsx
│   │   ├── layout.tsx
│   │   └── globals.css
│   ├── components/
│   │   └── ui/          # shadcn/ui components
│   ├── hooks/
│   │   ├── use-mobile.ts
│   │   └── use-toast.ts
│   └── lib/
│       ├── db.ts        # Prisma database client
│       ├── socket.ts    # Socket.IO configuration
│       └── utils.ts     # Utility functions
├── prisma/
│   └── schema.prisma    # Database schema
├── public/
│   ├── logo.svg
│   └── robots.txt
├── server.ts            # Custom server with Socket.IO
├── vercel.json          # Vercel configuration
├── next.config.ts       # Next.js configuration
├── tailwind.config.ts   # Tailwind CSS configuration
├── package.json
└── README.md
```

Quick Start

Want to get up and running quickly? Follow these steps:

1. **Clone and install:**

```
git clone <your-repo-url>
cd QRCode
pnpm install
```

2. **Set up database:**

```
pnpm run db:push
```

3. **Start development:**

```
pnpm dev
```

4. **Open browser:**
   Visit http://localhost:3003

5. **Create your first catalogue:**
   - Click "Create New Catalogue"
   - Add a title and description
   - Add products with images
   - Generate QR codes
   - Share with customers!

🤝 Contributing

We welcome contributions! Here's how to get started:

1. **Fork the repository**
2. **Create a feature branch:**

```
git checkout -b feature/amazing-feature
```

3. **Make your changes**
4. **Test thoroughly:**
   - Test all CRUD operations
   - Verify image uploads work
   - Check QR code generation
   - Test on mobile devices

5. **Commit your changes:**

```
git commit -m "Add amazing feature"
```

6. **Push to your branch:**

```
git push origin feature/amazing-feature
```

7. **Submit a pull request**

Development Guidelines

- Follow the existing code style
- Add tests for new features
- Update documentation as needed
- Ensure mobile responsiveness
- Test with different image formats

License

This project is open source and available under the MIT License.

Acknowledgments

- Next.js - The React framework
- Tailwind CSS - Utility-first CSS framework
- shadcn/ui - Beautiful UI components
- Prisma - Database toolkit
- Socket.IO - Real-time communication
- Neon - Serverless PostgreSQL
- Vercel - Deployment platform

Made with ♥ for the community