🚀 Deployment Guide

This guide covers various deployment options for the Audience Response System, from local network hosting to cloud deployment.

📋 Prerequisites

Before deployment, ensure you have:
- Node.js 18+ installed
- PostgreSQL database (local or cloud)
- Git for version control
- Domain name (for production deployments)

🏠 Local Network Deployment

Perfect for classrooms, offices, or events where you want to keep everything on-premises.

Step 1: Server Setup

```
# Clone the repository
git clone https://github.com/benny2744/ARS-BZ.git
cd ARS-BZ/app

# Install dependencies
yarn install

# Set up environment variables
cp .env.example .env
```

Step 2: Configure Environment

Edit `.env` file:

```
# Use your machine's IP address for network access
NEXTAUTH_URL="http://192.168.1.100:3000"  # Replace with your IP
DATABASE_URL="postgresql://username:password@localhost:5432/ars_db"
NEXTAUTH_SECRET="your-secure-random-secret"
NODE_ENV="production"
```

Step 3: Database Setup

```
# Initialize database
npx prisma migrate deploy
npx prisma db seed
```

Step 4: Build and Start

```
# Build for production
yarn build

# Start the server
yarn start
```

Step 5: Network Configuration

1. **Find your IP address**:
   - Windows: `ipconfig`
   - macOS/Linux: `ifconfig` or `ip addr show`

2. **Configure firewall**:
   - Allow port 3000 through your firewall
   - On macOS: System Preferences → Security & Privacy → Firewall
   - On Windows: Windows Defender Firewall settings

3. **Share the URL**: `http://[YOUR-IP]:3000`

☁️ Cloud Deployment

Vercel (Recommended for Next.js)

1. **Prepare your repository**:
   ```
   bash
      # Ensure your code is committed
      git add .
      git commit -m "Prepare for deployment"
      git push origin main
   ```

2. **Deploy to Vercel**:
   - Visit vercel.com (https://vercel.com)
   - Connect your GitHub repository
   - Configure environment variables
   - Deploy automatically

3. **Environment Variables**:
   Set these in Vercel dashboard:
   ```
   DATABASE_URL=postgresql://...
      NEXTAUTH_URL=https://your-app.vercel.app
      NEXTAUTH_SECRET=your-secret
      NODE_ENV=production
   ```

Railway

1. **Connect repository**:
   - Visit railway.app (https://railway.app)
   - Connect GitHub repository
   - Choose the `app` directory as root

2. **Add PostgreSQL database**:
   - Add PostgreSQL service in Railway
   - Note the connection URL

3. **Configure variables**:
   `DATABASE_URL=${{Postgres.DATABASE_URL}}`
   `NEXTAUTH_URL=${{RAILWAY_PUBLIC_DOMAIN}}`
   `NEXTAUTH_SECRET=your-secret`

DigitalOcean App Platform

1. **Create new app**:
   - Connect GitHub repository
   - Select Node.js environment
   - Set build command: `yarn build`
   - Set run command: `yarn start`

2. **Database setup**:
   - Create managed PostgreSQL database
   - Add connection string to environment

🐳 Docker Deployment

Create a `Dockerfile` in the app directory:

```
FROM node:18-alpine

WORKDIR /app

# Copy package files
COPY package.json yarn.lock ./

# Install dependencies
RUN yarn install --frozen-lockfile

# Copy source code
COPY . .

# Generate Prisma client
RUN npx prisma generate

# Build application
RUN yarn build

EXPOSE 3000

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

Create `docker-compose.yml`:

```yaml
version: '3.8'
services:
  app:
    build: ./app
    ports:
      - "3000:3000"
    environment:
      - DATABASE_URL=postgresql://ars:password@db:5432/ars_db
      - NEXTAUTH_URL=http://localhost:3000
      - NEXTAUTH_SECRET=your-secret
    depends_on:
      - db

  db:
    image: postgres:14
    environment:
      POSTGRES_DB: ars_db
      POSTGRES_USER: ars
      POSTGRES_PASSWORD: password
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  postgres_data:
```

Deploy with Docker:

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

🔧 Environment Configuration

Required Environment Variables

Variable	Description	Example
`DATABASE_URL`	PostgreSQL connection string	`postgresql://user:pass@host:5432/db`
`NEXTAUTH_URL`	Your app’s URL	`http://localhost:3000`
`NEXTAUTH_SECRET`	Random secret for NextAuth	Generate with `openssl rand -base64 32`

Optional Variables

Variable	Description	Default
`NODE_ENV`	Environment mode	`development`
`MAX_FILE_SIZE`	Max upload size in bytes	`10485760` (10MB)
`LOG_LEVEL`	Logging level	`info`

📊 Performance Optimization

Production Settings

1. **Enable caching**:

```
javascript
   // next.config.js
   module.exports = {
     poweredByHeader: false,
     compress: true,
     // ... other optimizations
   }
```

2. **Database optimization**:
   - Use connection pooling
   - Enable query optimization
   - Regular database maintenance

3. **CDN setup** (for static assets):
   - Upload images to a CDN
   - Configure next.config.js for external images

🔒 Security Checklist

Production Security

- [ ] Use HTTPS in production
- [ ] Set secure environment variables
- [ ] Configure CORS appropriately
- [ ] Enable database SSL
- [ ] Set up proper firewall rules
- [ ] Use strong secrets and passwords
- [ ] Regular security updates
- [ ] Monitor for suspicious activity

Network Security

- [ ] Disable unnecessary ports
- [ ] Use VPN for admin access (if needed)
- [ ] Configure rate limiting
- [ ] Enable logging and monitoring
- [ ] Regular backups

📈 Monitoring & Maintenance

Health Monitoring

Set up monitoring for:
- Application uptime
- Database connection
- File upload functionality

- Memory and CPU usage
- Error rates

Backup Strategy

1. **Database backups**:
   ```
   bash
      # Daily backup script
      pg_dump $DATABASE_URL > backup-$(date +%Y%m%d).sql
   ```

2. **File uploads**:
   - Backup uploads directory regularly
   - Consider cloud storage for uploads

3. **Application backups**:
   - Keep Git history
   - Tag releases for rollback capability

🆘 Troubleshooting

Common Issues

Database Connection Errors:
- Check DATABASE_URL format
- Verify database is running
- Check network connectivity

File Upload Issues:
- Verify uploads directory permissions
- Check disk space
- Review file size limits

Authentication Problems:
- Verify NEXTAUTH_URL matches deployment URL
- Check NEXTAUTH_SECRET is set
- Review callback URLs

Logs and Debugging

Enable detailed logging:

```
LOG_LEVEL=debug
NODE_ENV=development  # For detailed errors
```

Check logs:

```
# Docker logs
docker-compose logs -f app

# Local logs
yarn dev  # Development mode with detailed output
```

📞 Support

For deployment support:
- Check the troubleshooting section
- Review GitHub issues
- Create a new issue with deployment details
- Include environment information and logs

Happy Deploying! 🚀