Deployment Guide

This document provides detailed instructions for deploying the AI Humanizer application to production environments.

Table of Contents

Prerequisites

Before deploying the application, ensure you have the following:

- Git repository access
- Heroku account (for backend deployment)
- Netlify account (for frontend deployment)
- PostgreSQL database provider (e.g., Heroku Postgres, AWS RDS)
- Domain name (optional, but recommended for production)
- SSL certificate (Let's Encrypt or through your domain provider)
- Node.js and npm installed locally

Database Deployment

Option 1: Heroku Postgres

1. Log in to your Heroku account
2. Create a new Postgres database:

```
heroku addons:create heroku-postgresql:hobby-dev -a your-app-name
```

3. Get the database credentials:

```
heroku pg:credentials:url -a your-app-name
```

4. Note down the connection string for later use

Option 2: AWS RDS

1. Log in to AWS Console
2. Navigate to RDS service
3. Click "Create Database"
4. Select PostgreSQL as the engine
5. Choose the appropriate tier (t2.micro is sufficient for testing)
6. Configure storage, credentials, and network settings
7. Create the database
8. Note down the endpoint, port, username, password, and database name

Database Migration

1. Connect to your production database:

```
psql -U username -h hostname -d database_name -f database.sql
```

2. Alternatively, use Sequelize migrations:

```
npx sequelize-cli db:migrate
```

Backend API Deployment

Deploying to Heroku

1. Install Heroku CLI:

```
npm install -g heroku
```

2. Log in to Heroku:

```
heroku login
```

3. Create a new Heroku app:

```
heroku create ai-humanizer-api
```

4. Add Postgres add-on (if not already created):

```
heroku addons:create heroku-postgresql:hobby-dev
```

5. Configure environment variables:

```
heroku config:set JWT_SECRET=your-secure-jwt-secret
heroku config:set NODE_ENV=production
heroku config:set UNDETECTABLE_API_KEY=your-api-key
```

6. Add a Procfile to the project root:

```
web: npm run start:prod
```

7. Add the build script to package.json:

```
"scripts": {
  "start:prod": "node dist/server/index.js",
  "build": "tsc -p tsconfig.json"
}
```

8. Deploy to Heroku:

```
git push heroku main
```

9. Ensure the app is running:

```
heroku logs --tail
```

Deploying to AWS Elastic Beanstalk

1. Install AWS CLI and EB CLI
2. Initialize EB application:

```
eb init
```

3. Create an environment:

```
eb create ai-humanizer-api-prod
```

4. Configure environment variables through the AWS console
5. Deploy:

```
eb deploy
```

Frontend Deployment

Deploying to Netlify

1. Install Netlify CLI:

```
npm install -g netlify-cli
```

2. Log in to Netlify:

```
netlify login
```

3. Build the production frontend:

```
npm run build
```

4. Create a netlify.toml file in the project root:

```
[build]
  publish = "build"
  command = "npm run build"

[[redirects]]
  from = "/*"
  to = "/index.html"
  status = 200
```

5. Deploy to Netlify:

```
netlify deploy --prod
```

6. Alternatively, connect your GitHub repository to Netlify for automatic deployments

Deploying to Vercel

1. Install Vercel CLI:

```
npm install -g vercel
```

2. Log in to Vercel:

```
vercel login
```

3. Deploy to Vercel:

```
vercel --prod
```

Environment Configuration

Backend Production Configuration

Create a production .env file or set environment variables in your hosting platform:

```
NODE_ENV=production
PORT=3001
DB_HOST=your-production-db-host
DB_PORT=5432
DB_NAME=your-production-db-name
DB_USER=your-production-db-user
DB_PASSWORD=your-production-db-password
JWT_SECRET=your-secure-jwt-secret
UNDETECTABLE_API_KEY=your-api-key
CORS_ORIGIN=https://your-frontend-domain.com
```

Frontend Production Configuration

Create a production .env file:

```
REACT_APP_API_URL=https://your-api-domain.com
REACT_APP_ENV=production
```

Domain Setup and SSL

Backend API Domain

1. Purchase a domain (e.g., api.aihumanizer.com)
2. In Heroku:

```
heroku domains:add api.aihumanizer.com -a your-app-name
```

3. Add the provided DNS target to your domain DNS settings
4. Enable Automatic Certificate Management:

```
heroku certs:auto:enable -a your-app-name
```

Frontend Domain

1. Purchase a domain (e.g., aihumanizer.com)
2. In Netlify, go to Domain Management
3. Click "Add custom domain"
4. Enter your domain name
5. Follow the DNS configuration instructions
6. Enable HTTPS

Continuous Integration/Deployment

GitHub Actions for CI/CD

1. Create `.github/workflows/backend.yml` :

```yaml
name: Backend CI/CD

on:
  push:
    branches: [ main ]
    paths:
      - 'src/server/**'
      - 'package.json'
      - 'package-lock.json'

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Setup Node.js
        uses: actions/setup-node@v2
        with:
          node-version: '14'
      - name: Install dependencies
        run: npm ci
      - name: Run tests
        run: npm test
      - name: Deploy to Heroku
        uses: akhileshns/heroku-deploy@v3.12.12
        with:
          heroku_api_key: ${{ secrets.HEROKU_API_KEY }}
          heroku_app_name: "your-app-name"
          heroku_email: ${{ secrets.HEROKU_EMAIL }}
```

2. Create `.github/workflows/frontend.yml` :

```yaml
name: Frontend CI/CD

on:
  push:
    branches: [ main ]
    paths:
      - 'src/**'
      - '!src/server/**'
      - 'public/**'
      - 'package.json'
      - 'package-lock.json'

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Setup Node.js
        uses: actions/setup-node@v2
        with:
          node-version: '14'
      - name: Install dependencies
        run: npm ci
      - name: Build
        run: npm run build
      - name: Deploy to Netlify
        uses: netlify/actions/cli@master
        with:
          args: deploy --dir=build --prod
        env:
          NETLIFY_AUTH_TOKEN: ${{ secrets.NETLIFY_AUTH_TOKEN }}
          NETLIFY_SITE_ID: ${{ secrets.NETLIFY_SITE_ID }}
```

Monitoring and Maintenance

Backend Monitoring

1. Set up Heroku application metrics:

```
heroku addons:create librato:development
```

2. Set up error tracking with Sentry:

```
heroku addons:create sentry:f1
```

3. Regular database backups:

```
heroku pg:backups:schedule DATABASE_URL --at '02:00 America/New_York' --app your-app-name
```

Frontend Monitoring

1. Set up Google Analytics in your React application
2. Configure Netlify Analytics from your Netlify dashboard
3. Implement error tracking with Sentry.io:

```
import * as Sentry from '@sentry/react';

Sentry.init({
  dsn: "your-sentry-dsn",
  environment: process.env.REACT_APP_ENV
});
```

Regular Maintenance Tasks

1. Update dependencies monthly:

```
npm outdated
npm update
```

2. Review and rotate JWT secrets quarterly
3. Monitor database performance and optimize queries
4. Set up automated security scans for vulnerabilities
5. Perform regular database backups
6. Monitor API usage and rate limits with Undetectable AI

Scaling Considerations

Backend Scaling

1. Configure Heroku auto-scaling:

```
heroku autoscale:enable web --min=1 --max=5 --p95=50ms --app your-app-name
```

2. Implement Redis caching for frequent queries:

```
heroku addons:create heroku-redis:hobby-dev
```

Database Scaling

1. Upgrade PostgreSQL plan as needed
2. Implement database connection pooling
3. Consider read replicas for heavy read operations

Frontend Performance

1. Implement code splitting and lazy loading
2. Optimize image sizes and formats
3. Use a Content Delivery Network (CDN) for static assets
4. Implement server-side rendering for improved SEO and performance

Backup and Disaster Recovery

1. Schedule regular database backups
2. Store configuration in version control
3. Document recovery procedures
4. Implement health checks and automated recovery
5. Test disaster recovery procedures quarterly