# New Stars Radio Ad Server

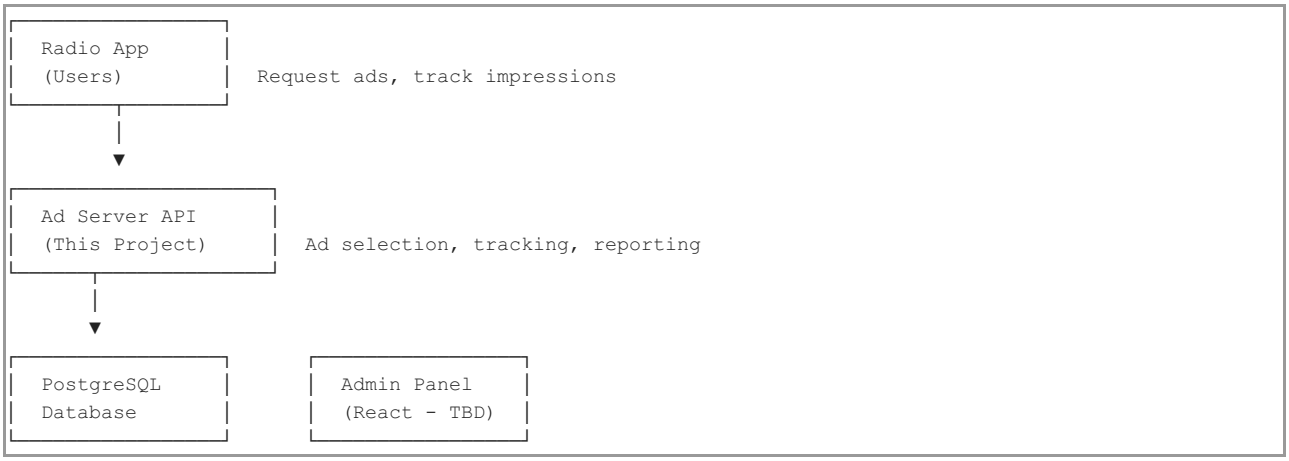
Professional ad management system for serving image ads in the New Stars Radio mobile application.

## 📋 Overview

This ad server enables you to:

- Serve targeted image ads to mobile radio listeners
- Track impressions and clicks with detailed analytics
- Manage campaigns through an admin panel
- Generate revenue from local advertisers
- Fallback to AdSense when no direct ads available

## 🏗️ Architecture




## ✨ Features (QS-Prompt 1 Complete)

### ✅ Backend Foundation

- FastAPI application with async support
- PostgreSQL database with SQLAlchemy ORM
- Complete database schema with indexes
- Alembic migrations
- JWT authentication system
- Password hashing with bcrypt
- Environment-based configuration
- Docker deployment ready

### ✅ Database Models

- Users (admin authentication)
- Advertisers
- Campaigns (with targeting & budgets)
- Ad Creatives (image ads)
- Impressions (tracking)
- Clicks (tracking)

### ✅ Security

- JWT token generation & validation
- Password hashing
- CORS middleware
- Input validation with Pydantic

## 🚀 Quick Start

### Prerequisites

- Docker and Docker Compose

- Git

### Setup

1. **Clone and navigate to the project:**

```
cd "D:\MUSIC - COMEDY\New New Stars Radio App\ad-server"
```

2. **Copy environment file:**

```
# Copy env.example to .env and configure
copy env.example .env
```

3. **Start services with Docker Compose:**

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

4. **Wait for services to be healthy (30 seconds)**

5. **Run database migrations:**

```
docker-compose exec backend alembic upgrade head
```

6. **Seed initial admin user:**

```
docker-compose exec backend python -m app.db.seed
```

7. **Access the application:**

- API: http://localhost:8000
- API Docs: http://localhost:8000/docs
- Health Check: http://localhost:8000/health

### Default Credentials

```
Email: admin@newstarsradio.com
Password: changeme123
```

⚠ **CHANGE THIS PASSWORD IMMEDIATELY IN PRODUCTION!**

## 🗀 Project Structure

```
ad-server/
├── app/
│   ├── api/
│   │   └── v1/
│   │       ├── endpoints/        # API route handlers (to be added)
│   │       └── router.py         # Main API router
│   ├── core/
│   │   ├── config.py            # Application settings
│   │   ├── database.py          # Database connection
│   │   └── security.py          # JWT & password utilities
│   ├── models/                  # SQLAlchemy models
│   │   ├── user.py
│   │   ├── advertiser.py
│   │   ├── campaign.py
│   │   ├── ad_creative.py
│   │   ├── impression.py
│   │   └── click.py
│   ├── schemas/                 # Pydantic schemas (to be added)
│   ├── services/                # Business logic (to be added)
│   ├── crud/                    # Database operations (to be added)
│   ├── db/
│   │   └── seed.py             # Database seeding
│   └── main.py                  # FastAPI application
├── alembic/
│   ├── versions/
│   │   └── 20251119_initial_schema.py
│   └── env.py
├── static/
│   └── ads/                     # Uploaded ad images
├── logs/                        # Application logs
├── tests/                       # Tests (to be added)
├── docker-compose.yml           # Development environment
├── Dockerfile                   # Backend container
├── requirements.txt             # Python dependencies
├── alembic.ini                 # Alembic configuration
└── README.md                    # This file
```

## 🗄️ Database Schema

### Users

- Admin and sales rep authentication
- Role-based access control

### Advertisers

- Company/individual information
- Contact details

### Campaigns

- Date ranges and budgets
- Priority levels (1-10)
- Geographic targeting (cities, states)
- Impression tracking

### Ad Creatives

- Image details (URL, dimensions)
- Click destinations
- Multiple creatives per campaign

### Impressions & Clicks

- User tracking
- Location data
- Timestamps for analytics

## 🔧 Configuration

Key environment variables (see `env.example`):

```
# Database
DATABASE_URL=postgresql://postgres:postgres@postgres:5432/adserver_dev

# Security
SECRET_KEY=your-secret-key-32-characters-minimum
ACCESS_TOKEN_EXPIRE_MINUTES=60

# CORS
CORS_ORIGINS=http://localhost:3000,http://localhost:5173

# File Upload
MAX_UPLOAD_SIZE=5242880  # 5MB
ALLOWED_EXTENSIONS=.jpg,.jpeg,.png,.gif,.webp
```

## Development

### View Logs

```
docker-compose logs -f
docker-compose logs -f backend  # Backend only
```

### Access Database

```
docker-compose exec postgres psql -U postgres -d adserver_dev
```

### Run Migrations

```
# Create new migration
docker-compose exec backend alembic revision --autogenerate -m "description"

# Apply migrations
docker-compose exec backend alembic upgrade head

# Rollback one version
docker-compose exec backend alembic downgrade -1
```

### Stop Services

```
docker-compose down
```

### Reset Database (⚠ Destroys all data)

```
docker-compose down -v
docker-compose up -d
docker-compose exec backend alembic upgrade head
docker-compose exec backend python -m app.db.seed
```

## API Documentation

Once running, visit:

- **Interactive Docs**: http://localhost:8000/docs
- **ReDoc**: http://localhost:8000/redoc
- **OpenAPI JSON**: http://localhost:8000/api/v1/openapi.json

## User Guide

**New to the Ad Manager?** Check out the comprehensive user guide:

- [Ad Manager User Guide (./AD_MANAGER_GUIDE.md)] - Complete guide on how to use the ad manager, including step-by-step instructions for managing advertisers, campaigns, and creatives.

## Security Features

- ✅ JWT authentication with expiry
- ✅ Password hashing with bcrypt

- ✅ CORS configuration
- ✅ Input validation (Pydantic)
- ✅ SQL injection protection (SQLAlchemy)
- ✅ Non-root Docker user
- ✅ Health check endpoint

## Next Steps (QS-Prompts 2-10)

The following features will be added in subsequent prompts:

**QS-Prompt 2**: Ad Selection & Tracking Services

- Ad selection algorithm
- Impression/click tracking
- Token validation

**QS-Prompt 3**: Ad Serving & Tracking APIs

- GET ad endpoint
- Track impression endpoint
- Track click endpoint
- Rate limiting

**QS-Prompt 4**: Campaign Management APIs

- CRUD operations for campaigns
- Creative upload & management
- Image processing

**QS-Prompt 5-7**: Admin Panel (React)

- Authentication UI
- Campaign management interface
- Reporting dashboard

**QS-Prompt 8-10**: Production Features

- Complete authentication flow
- Docker production config
- Testing & documentation

## Troubleshooting

### Port Already in Use

```
# Change port in docker-compose.yml
ports:
  - "8001:8000"  # Use 8001 instead
```

### Database Connection Failed

```
# Check if PostgreSQL is running
docker-compose ps

# Restart services
docker-compose restart postgres backend
```

### Migrations Failed

```
# Check current version
docker-compose exec backend alembic current

# View migration history
docker-compose exec backend alembic history

# Try manual migration
docker-compose exec backend alembic upgrade head --sql
```

## TODO

- [ ] Complete QS-Prompts 2-10
- [ ] Add API endpoints (ads, campaigns, tracking)
- [ ] Build admin panel (React)
- [ ] Add comprehensive tests
- [ ] Production deployment guide
- [ ] Mobile app integration examples

## License

MIT License - See LICENSE file for details

## Support

For issues and questions:

- GitHub Issues: [Create issue]
- Email: support@newstarsradio.com

---

**Status**: ✅ QS-Prompt 1 Complete - Backend Foundation Ready

**Next**: Run QS-Prompt 2 to add ad selection and tracking services