✅ QS-Prompt 4: COMPLETE - Campaign Management Admin Panel

Status: All Features Already Implemented ✅

Date: December 12, 2025
Prompt: QS-Prompt 4 - Campaign Management Admin Panel API

🎉 Discovery: Already Complete!

All admin panel endpoints were created in an earlier setup iteration and are fully functional!

What Exists

1. Authentication ✅

File: `app/api/v1/endpoints/auth.py`

Endpoints:

- ✅ `POST /api/v1/auth/login` - Admin login with JWT
- ✅ `GET /api/v1/auth/me` - Get current user info

Features:

- JWT token generation
- Password verification with bcrypt
- User role checking
- Active status validation

2. Advertiser CRUD ✅

File: `app/api/v1/endpoints/advertisers.py`

Endpoints:

- ✅ `POST /api/v1/advertisers` - Create advertiser
- ✅ `GET /api/v1/advertisers` - List all advertisers (paginated)
- ✅ `GET /api/v1/advertisers/{id}` - Get single advertiser
- ✅ `PUT /api/v1/advertisers/{id}` - Update advertiser
- ✅ `DELETE /api/v1/advertisers/{id}` - Delete advertiser

Features:

- Email uniqueness validation
- Status management (active/inactive)
- Company info tracking

- Pagination support (skip/limit)

3. Campaign CRUD ✅

File: `app/api/v1/endpoints/campaigns.py`

Endpoints:

- ✅ `POST /api/v1/campaigns` - Create campaign
- ✅ `GET /api/v1/campaigns` - List campaigns (filtered by advertiser, status)
- ✅ `GET /api/v1/campaigns/{id}` - Get single campaign
- ✅ `PUT /api/v1/campaigns/{id}` - Update campaign
- ✅ `DELETE /api/v1/campaigns/{id}` - Delete campaign

Features:

- Advertiser validation
- Date range validation (start < end)
- Priority validation (1-10)
- Budget management
- Geographic targeting (cities, states as JSON)
- Status management (draft/active/paused/completed)
- Cascading deletes (removes associated creatives)

4. Creative CRUD with Image Upload ✅

File: `app/api/v1/endpoints/creatives.py`

Endpoints:

- ✅ `POST /api/v1/creatives` - Create creative with image upload
- ✅ `GET /api/v1/creatives` - List creatives (filtered by campaign)
- ✅ `GET /api/v1/creatives/{id}` - Get single creative
- ✅ `PUT /api/v1/creatives/{id}` - Update creative
- ✅ `POST /api/v1/creatives/{id}/upload` - Upload/replace image
- ✅ `DELETE /api/v1/creatives/{id}` - Delete creative

Features:

- Multipart form-data image upload
- Image file validation (type, size)
- Automatic image dimension detection (using Pillow)
- File storage in `static/ads/`
- Campaign validation
- Status management (active/inactive)
- Click URL validation

5. Reporting & Analytics ✅

File: `app/api/v1/endpoints/reports.py`

Endpoints:

- ✅ `GET /api/v1/reports/campaigns/{id}/stats` - Campaign statistics
- ✅ `GET /api/v1/reports/creatives/{id}/stats` - Creative statistics
- ✅ `GET /api/v1/reports/overview` - Overall performance

Features:

- Date range filtering
- Impression counts
- Click counts
- CTR calculation (Click-Through Rate)
- Budget utilization
- Geographic breakdown
- Time-series data

6. Security & Auth Dependencies ✅

File: `app/api/dependencies.py`

Functions:

- ✅ `get_current_user()` - Extract user from JWT token
- ✅ `get_current_admin()` - Require admin role

Features:

- Bearer token authentication
- JWT validation
- Role-based access control (RBAC)
- User active status checking

API Structure

Authentication Required

All admin endpoints require JWT authentication:

```
# 1. Login to get token
curl -X POST http://localhost:8000/api/v1/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"admin@newstarsradio.com","password":"changeme123"}'

# Response: {"access_token": "eyJ...", "token_type": "bearer"}

# 2. Use token in subsequent requests
curl -X GET http://localhost:8000/api/v1/advertisers \
  -H "Authorization: Bearer eyJ..."
```

Complete API Map

```
/api/v1/
├── auth/
│   ├── POST   /login             # Login
│   └── GET    /me                # Current user
│
├── advertisers/
│   ├── POST   /                  # Create
│   ├── GET    /                  # List (paginated)
│   ├── GET    /{id}              # Get one
│   ├── PUT    /{id}              # Update
│   └── DELETE /{id}              # Delete
│
├── campaigns/
│   ├── POST   /                  # Create
│   ├── GET    /                  # List (filtered)
│   ├── GET    /{id}              # Get one
│   ├── PUT    /{id}              # Update
│   └── DELETE /{id}              # Delete
│
├── creatives/
│   ├── POST   /                  # Create with image
│   ├── GET    /                  # List (filtered)
│   ├── GET    /{id}              # Get one
│   ├── PUT    /{id}              # Update
│   ├── POST   /{id}/upload       # Upload/replace image
│   └── DELETE /{id}              # Delete
│
├── reports/
│   ├── GET    /campaigns/{id}/stats    # Campaign stats
│   ├── GET    /creatives/{id}/stats    # Creative stats
│   └── GET    /overview                # Overall stats
│
└── ads/
    ├── POST   /request                 # Get ad to display
    └── tracking/
        ├── POST /impression            # Track impression
        ├── POST /click                 # Track click
        └── GET  /click/{token}         # Click redirect
```

Example Usage

1. Login as Admin

```
curl -X POST http://localhost:8000/api/v1/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "admin@newstarsradio.com",
    "password": "changeme123"
  }'
```

Response:

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "bearer"
}
```

2. Create an Advertiser

```
curl -X POST http://localhost:8000/api/v1/advertisers \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "John Doe",
    "email": "john@example.com",
    "phone": "+1234567890",
    "company_name": "ACME Corp"
  }'
```

3. Create a Campaign

```
curl -X POST http://localhost:8000/api/v1/campaigns \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "advertiser_id": "550e8400-e29b-41d4-a716-446655440000",
    "name": "Summer Sale 2025",
    "start_date": "2025-06-01T00:00:00Z",
    "end_date": "2025-08-31T23:59:59Z",
    "priority": 8,
    "impression_budget": 100000,
    "target_cities": ["New York", "Los Angeles"],
    "target_states": ["NY", "CA"]
  }'
```

4. Upload Ad Creative

```
curl -X POST http://localhost:8000/api/v1/creatives \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -F "campaign_id=660e8400-e29b-41d4-a716-446655440001" \
  -F "name=Banner Ad 728x90" \
  -F "click_url=https://example.com/sale" \
  -F "alt_text=Check out our summer sale!" \
  -F "image_file=@banner-728x90.jpg"
```

5. Get Campaign Statistics

```
curl -X GET "http://localhost:8000/api/v1/reports/campaigns/660e8400-e29b-41d4-a716-446655440001/stats?start_date=2025-06-01T00:00:00Z" \
  -H "Authorization: Bearer YOUR_TOKEN"
```

Response:

```
{
  "campaign_id": "660e8400-e29b-41d4-a716-446655440001",
  "campaign_name": "Summer Sale 2025",
  "impressions": 45230,
  "clicks": 892,
  "ctr": 1.97,
  "budget_used": 45.23,
  "budget_total": 100.00
}
```

Testing the Admin Panel

1. Via Swagger UI (Recommended)

1. Start services: `docker compose up -d`
2. Open: http://localhost:8000/docs
3. Click **"Authorize"** button (top right)
4. Login via `/api/v1/auth/login` endpoint
5. Copy the `access_token` from response
6. Paste into Authorization dialog: `Bearer <token>`
7. Now you can test all endpoints!

2. Via cURL

See examples above - replace `YOUR_TOKEN` with actual JWT from login.

3. Via Postman/Insomnia

1. Import OpenAPI spec: http://localhost:8000/openapi.json
2. Set Authorization: Bearer Token
3. Use token from login response

Database Schemas

All schemas already exist:

Pydantic Schemas:

- ✅ `app/schemas/advertiser.py` - AdvertiserCreate, AdvertiserUpdate, AdvertiserResponse
- ✅ `app/schemas/campaign.py` - CampaignCreate, CampaignUpdate, CampaignResponse
- ✅ `app/schemas/creative.py` - CreativeCreate, CreativeUpdate, CreativeResponse
- ✅ `app/schemas/report.py` - CampaignStats, CreativeStats, OverviewStats
- ✅ `app/schemas/auth.py` - LoginRequest, Token, UserResponse

Database Models:

- ✅ `app/models/user.py` - Admin users
- ✅ `app/models/advertiser.py` - Advertisers
- ✅ `app/models/campaign.py` - Campaigns
- ✅ `app/models/ad_creative.py` - Ad creatives
- ✅ `app/models/impression.py` - Impressions
- ✅ `app/models/click.py` - Clicks

Security Features

1. ✅ **JWT Authentication** - All admin endpoints protected
2. ✅ **Password Hashing** - Bcrypt for secure password storage
3. ✅ **Role-Based Access** - Admin role required for management endpoints
4. ✅ **Input Validation** - Pydantic schemas validate all inputs
5. ✅ **SQL Injection Protection** - SQLAlchemy ORM prevents SQL injection
6. ✅ **CORS Configuration** - Configurable allowed origins
7. ✅ **Rate Limiting** - Middleware prevents abuse (from QS-Prompt 3)

File Upload Configuration

Upload Directory: `static/ads/`
Supported Formats: JPEG, PNG, GIF, WebP
Max File Size: 5MB (configurable)
Filename Pattern: `{campaign_id}_{original_filename}`

Image Processing:

- Automatic dimension detection (width × height)
- Validation of file types
- Storage in static directory for serving

TODO List Status: 6/6 Complete ✅

1. ✅ Create admin authentication schemas and endpoints

2. ✅ Build Advertiser CRUD endpoints
3. ✅ Build Campaign CRUD endpoints
4. ✅ Build Creative CRUD endpoints with image upload
5. ✅ Add basic reporting endpoints
6. ✅ Update existing endpoint schemas

Next Steps

QS-Prompt 5: Frontend React Admin Panel

Will create:

- React 18 + TypeScript frontend
- Login page with JWT storage
- Advertiser management UI
- Campaign management UI (with date pickers, targeting)
- Creative management UI (with image upload)
- Dashboard with reports/charts
- Routing with React Router v6
- State management with Zustand
- API client with TanStack Query

Status: ✅ **QS-Prompt 4 COMPLETE - ALL FEATURES EXIST**
API Endpoints: 20+ endpoints fully functional
Database: All tables and relationships in place
Security: JWT auth + RBAC implemented
Ready for: QS-Prompt 5 (Frontend) 🚀