

# Analytics Service API Implementation Summary

---

## Overview

Complete implementation of Analytics Service Phase 1 with service layers and API endpoints for aggregating, analyzing, and reporting data from all Mission Engadi platform services.

**Generated:** December 24, 2025

**Service Port:** 8009

**Base URL:** <http://localhost:8009/api/v1>

---

## Architecture Components

### 1. Service Integration Layer

#### HTTP Client ( `app/core/service_client.py` )

- **ServiceClient:** Async HTTP client with retry logic
- Configurable timeout and max retries
- Exponential backoff for failed requests
- Context manager support
- Authentication token handling
- **ServiceURLs:** URL builder for microservices
  - Auth Service (Port 8002)
  - Content Service (Port 8003)
  - Partners CRM Service (Port 8005)
  - Projects Service (Port 8006)
  - Social Media Service (Port 8007)
  - Notification Service (Port 8008)

## 2. Configuration Updates

### New Settings ( app/core/config.py )

```
# Microservices URLs
AUTH_SERVICE_URL: str = "http://localhost:8002"
CONTENT_SERVICE_URL: str = "http://localhost:8003"
PARTNERS_CRM_SERVICE_URL: str = "http://localhost:8005"
PROJECTS_SERVICE_URL: str = "http://localhost:8006"
SOCIAL_MEDIA_SERVICE_URL: str = "http://localhost:8007"
NOTIFICATION_SERVICE_URL: str = "http://localhost:8008"

# Data Synchronization
SYNC_ENABLED: bool = True
SYNC_INTERVAL_MINUTES: int = 60
SYNC_BATCH_SIZE: int = 1000
SYNC_TIMEOUT_SECONDS: int = 300
SYNC_MAX_RETRIES: int = 3

# Cache Settings
CACHE_ENABLED: bool = True
CACHE_TTL_SECONDS: int = 3600
CACHE_METRICS_TTL_SECONDS: int = 300
CACHE_DASHBOARD_TTL_SECONDS: int = 600

# Analytics Settings
ANALYTICS_RETENTION_DAYS: int = 1095 # 3 years
ANALYTICS_AGGREGATION_LEVELS: List[str] = ["hourly", "daily", "weekly", "monthly"]
```

## Service Layers (8 Services)

### 1. MetricService ( app/services/metric\_service.py )

**Purpose:** Core metric operations and aggregations

**Methods:**

- `create_metric()` - Create new metric
- `get_metric()` - Get by ID
- `list_metrics()` - List with filters
- `delete_metric()` - Delete metric
- `aggregate_metrics()` - Aggregate with statistics
- `get_metrics_by_service()` - Filter by service
- `get_metrics_by_type()` - Filter by type
- `get_time_series()` - Time-series data

**Features:**

- Complex filtering (service, type, name, date range)
- Statistical aggregations (sum, avg, min, max, count)
- Time-series data generation
- Efficient indexing support

### 2. DashboardService ( app/services/dashboard\_service.py )

**Purpose:** Dashboard management and data fetching

**Methods:**

- `create_dashboard()` - Create dashboard
- `get_dashboard()` - Get by ID
- `list_dashboards()` - List with filters
- `update_dashboard()` - Update dashboard
- `delete_dashboard()` - Delete dashboard
- `get_dashboard_data()` - Get with widget data
- `get_executive_dashboard()` - Executive dashboard

**Features:**

- CRUD operations for dashboards
- Widget configuration support
- Access control (public/private, default)
- Dashboard type filtering

### **3. DataSyncService ( `app/services/data_sync_service.py` )**

**Purpose:** Sync job tracking and management

**Methods:**

- `create_sync_record()` - Create sync record
- `get_sync_record()` - Get by ID
- `list_sync_records()` - List with filters
- `update_sync_record()` - Update record
- `get_sync_status()` - Current sync status
- `get_sync_statistics()` - Sync statistics

**Features:**

- Sync job tracking
- Status monitoring
- Error logging
- Performance metrics

### **4. PartnerAnalyticsService ( `app/services/partner_analytics_service.py` )**

**Purpose:** Partner data analytics

**Methods:**

- `get_partner_statistics()` - Partner stats
- `get_donation_trends()` - Donation trends
- `get_engagement_metrics()` - Engagement metrics
- `get_partner_breakdown()` - Partner type breakdown

**Integration:**

- Fetches from Partners CRM Service (Port 8005)
- Fallback to local metrics
- Aggregates donation data
- Calculates engagement rates

### **5. ProjectAnalyticsService ( `app/services/project_analytics_service.py` )**

**Purpose:** Project data analytics

**Methods:**

- `get_project_statistics()` - Project stats
- `get_impact_metrics()` - Impact metrics
- `get_completion_rates()` - Completion rates
- `get_beneficiary_trends()` - Beneficiary trends

**Integration:**

- Fetches from Projects Service (Port 8006)
- Calculates impact metrics
- Tracks beneficiaries
- Monitors project completion

**6. SocialMediaAnalyticsService ( app/services/social\_media\_analytics\_service.py )****Purpose:** Social media analytics**Methods:**

- `get_performance_metrics()` - Performance metrics
- `get_platform_comparison()` - Platform comparison
- `get_engagement_trends()` - Engagement trends

**Integration:**

- Aggregates from Social Media Service (Port 8007)
- Cross-platform analysis
- Engagement tracking
- Post performance metrics

**7. NotificationAnalyticsService ( app/services/notification\_analytics\_service.py )****Purpose:** Notification analytics**Methods:**

- `get_notification_statistics()` - Notification stats
- `get_delivery_rates()` - Delivery rates
- `get_channel_effectiveness()` - Channel effectiveness

**Integration:**

- Fetches from Notification Service (Port 8008)
- Delivery rate calculation
- Channel comparison
- Campaign performance

**8. AggregationService ( app/services/aggregation\_service.py )****Purpose:** Orchestrate data aggregation from all services**Methods:**

- `trigger_sync()` - Trigger sync for service
- `aggregate_all_services()` - Aggregate all services
- `_sync_partners_data()` - Sync Partners CRM
- `_sync_projects_data()` - Sync Projects
- `_sync_social_media_data()` - Sync Social Media
- `_sync_notification_data()` - Sync Notifications

**Features:**

- Orchestrates multi-service sync
  - Error handling and retries
  - Batch processing
  - Progress tracking
- 

## API Endpoints (35 Total)

### Metrics Endpoints (8 endpoints)

**Base:** /api/v1/metrics

1. **POST** / - Create metric
  - **Auth:** Required
  - **Body:** MetricCreate schema
  - **Response:** MetricResponse (201)
  
2. **GET** /{metric\_id} - Get metric by ID
  - **Auth:** None
  - **Response:** MetricResponse
  
3. **GET** / - List metrics
  - **Auth:** None
  - **Filters:** service\_name, metric\_type, metric\_name, start\_date, end\_date
  - **Pagination:** skip, limit
  - **Response:** List[MetricResponse]
  
4. **DELETE** /{metric\_id} - Delete metric
  - **Auth:** Required
  - **Response:** 204 No Content
  
5. **GET** /aggregate/statistics - Aggregate metrics
  - **Auth:** Required
  - **Filters:** service\_name, metric\_type, metric\_name, dates, group\_by
  - **Response:** List[MetricAggregation]
  
6. **GET** /by-service/{service\_name} - Get by service
  - **Auth:** None
  - **Filters:** start\_date, end\_date, limit
  - **Response:** List[MetricResponse]
  
7. **GET** /by-type/{metric\_type} - Get by type
  - **Auth:** None
  - **Filters:** start\_date, end\_date, limit
  - **Response:** List[MetricResponse]
  
8. **GET** /time-series/data - Get time-series data
  - **Auth:** None
  - **Filters:** service\_name, metric\_type, metric\_name, dates, interval
  - **Response:** Time-series data

## Dashboard Endpoints (7 endpoints)

**Base:** /api/v1/dashboards

1. **POST** / - Create dashboard
  - **Auth:** Required
  - **Body:** DashboardCreate schema
  - **Response:** DashboardResponse (201)
  
2. **GET** /{dashboard\_id} - Get dashboard by ID
  - **Auth:** None
  - **Response:** DashboardResponse
  
3. **GET** / - List dashboards
  - **Auth:** None
  - **Filters:** dashboard\_type, is\_default, is\_public
  - **Pagination:** skip, limit
  - **Response:** List[DashboardResponse]
  
4. **PUT** /{dashboard\_id} - Update dashboard
  - **Auth:** Required
  - **Body:** DashboardUpdate schema
  - **Response:** DashboardResponse
  
5. **DELETE** /{dashboard\_id} - Delete dashboard
  - **Auth:** Required
  - **Response:** 204 No Content
  
6. **GET** /{dashboard\_id}/data - Get dashboard data
  - **Auth:** None
  - **Response:** Dashboard data with widgets
  
7. **GET** /executive/default - Get executive dashboard
  - **Auth:** None
  - **Response:** Executive dashboard data

## Data Sync Endpoints (6 endpoints)

**Base:** /api/v1-sync

1. **POST** / - Trigger manual sync
  - **Auth:** Required
  - **Query:** service\_name, sync\_type
  - **Response:** Sync result
  
2. **GET** /{sync\_id} - Get sync record
  - **Auth:** None
  - **Response:** DataSyncResponse
  
3. **GET** / - List sync records
  - **Auth:** None
  - **Filters:** service\_name, sync\_type, status
  - **Pagination:** skip, limit
  - **Response:** List[DataSyncResponse]

4. **GET** /status/current - Get sync status
  - **Auth:** None
  - **Filters:** service\_name
  - **Response:** Current sync status
  
5. **GET** /statistics/summary - Get sync statistics
  - **Auth:** Required
  - **Filters:** service\_name
  - **Response:** Sync statistics
  
6. **POST** /aggregate-all - Aggregate all services
  - **Auth:** Required
  - **Response:** Aggregation results

## Partner Analytics Endpoints (4 endpoints)

**Base:** /api/v1/analytics/partners

1. **GET** /statistics - Get partner statistics
  - **Auth:** None
  - **Filters:** start\_date, end\_date
  - **Response:** Partner statistics
  
2. **GET** /donations - Get donation trends
  - **Auth:** None
  - **Filters:** start\_date, end\_date
  - **Response:** Donation trends
  
3. **GET** /engagement - Get engagement metrics
  - **Auth:** None
  - **Filters:** start\_date, end\_date
  - **Response:** Engagement metrics
  
4. **GET** /breakdown - Get partner breakdown
  - **Auth:** None
  - **Response:** Partner type breakdown

## Project Analytics Endpoints (4 endpoints)

**Base:** /api/v1/analytics/projects

1. **GET** /statistics - Get project statistics
  - **Auth:** None
  - **Filters:** start\_date, end\_date
  - **Response:** Project statistics
  
2. **GET** /impact - Get impact metrics
  - **Auth:** None
  - **Filters:** start\_date, end\_date
  - **Response:** Impact metrics
  
3. **GET** /completion - Get completion rates
  - **Auth:** None
  - **Filters:** start\_date, end\_date
  - **Response:** Completion rates

4. **GET** /beneficiaries - Get beneficiary trends
  - **Auth:** None
  - **Filters:** start\_date, end\_date
  - **Response:** Beneficiary trends

## Social Media Analytics Endpoints (3 endpoints)

**Base:** /api/v1/analytics/social-media

1. **GET** /performance - Get performance metrics
  - **Auth:** None
  - **Filters:** start\_date, end\_date
  - **Response:** Performance metrics
2. **GET** /platforms - Get platform comparison
  - **Auth:** None
  - **Filters:** start\_date, end\_date
  - **Response:** Platform comparison
3. **GET** /engagement - Get engagement trends
  - **Auth:** None
  - **Filters:** start\_date, end\_date
  - **Response:** Engagement trends

## Notification Analytics Endpoints (3 endpoints)

**Base:** /api/v1/analytics/notifications

1. **GET** /statistics - Get notification statistics
  - **Auth:** None
  - **Filters:** start\_date, end\_date
  - **Response:** Notification statistics
2. **GET** /delivery - Get delivery rates
  - **Auth:** None
  - **Filters:** start\_date, end\_date
  - **Response:** Delivery rates
3. **GET** /channels - Get channel effectiveness
  - **Auth:** None
  - **Filters:** start\_date, end\_date
  - **Response:** Channel effectiveness

## API Dependencies

### Database Session ( get\_db )

- Provides async database session
- Automatic session cleanup
- Used by all endpoints

### Authentication ( get\_current\_user )

- Bearer token authentication

- Returns user information
  - Required for write operations
  - TODO: Integrate with Auth Service JWT validation
- 

## Testing the API

### 1. Start the Service

```
cd /home/ubuntu/analytics_service
uvicorn app.main:app --host 0.0.0.0 --port 8009 --reload
```

### 2. Access API Documentation

- **Swagger UI:** <http://localhost:8009/docs>
- **ReDoc:** <http://localhost:8009/redoc>
- **OpenAPI JSON:** <http://localhost:8009/openapi.json>

### 3. Health Check

```
curl http://localhost:8009/api/v1/health
```

### 4. Create a Metric (Example)

```
curl -X POST http://localhost:8009/api/v1/metrics \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_TOKEN" \
-d '{
  "service_name": "partners_crm",
  "metric_type": "donation",
  "metric_name": "monthly_donation",
  "value": 1000.0,
  "dimensions": {"partner_id": "123", "currency": "USD"},
  "timestamp": "2025-12-24T10:00:00Z",
  "date": "2025-12-24"
}'
```

### 5. List Metrics

```
curl "http://localhost:8009/api/v1/metrics?service_name=partners_crm&limit=10"
```

### 6. Get Partner Statistics

```
curl "http://localhost:8009/api/v1/analytics/partners/statistics?
start_date=2025-01-01&end_date=2025-12-24"
```

## 7. Trigger Sync

```
curl -X POST "http://localhost:8009/api/v1/sync?service_name=partners_crm&sync_type=manual" \
-H "Authorization: Bearer YOUR_TOKEN"
```

## Error Handling

All endpoints follow standard HTTP status codes:

- **200 OK** - Successful GET/PUT
- **201 Created** - Successful POST
- **204 No Content** - Successful DELETE
- **400 Bad Request** - Invalid input
- **401 Unauthorized** - Authentication required
- **404 Not Found** - Resource not found
- **500 Internal Server Error** - Server error

Error Response Format:

```
{
  "detail": "Error message"
}
```

## Performance Considerations

### Indexing

- All date-based queries use indexed fields
- Composite indexes for common query patterns
- 20+ indexes across 3 tables

### Pagination

- Default limit: 100 records
- Maximum limit: 1000 records
- Skip/limit pagination support

### Caching (Future)

- Redis integration ready
- Configurable TTL per resource type
- Cache invalidation on updates

### Aggregation

- Batch processing (1000 records/batch)
- Timeout protection (300 seconds)
- Retry logic with exponential backoff

# Security

---

## Authentication

- Bearer token authentication
- Middleware integration ready
- Protected write operations

## Authorization (Future)

- Role-based access control
  - Resource-level permissions
  - Service-to-service auth
- 

# Future Enhancements

---

## Phase 2 Features

1. Real-time analytics with WebSockets
2. Advanced ML-powered predictions
3. Custom report generation
4. Export functionality (PDF, Excel)
5. Scheduled report delivery
6. Alert system integration

## Optimization

1. Implement Redis caching
  2. Add query result caching
  3. Optimize aggregation queries
  4. Add database connection pooling
  5. Implement rate limiting
- 

# Files Created/Modified

---

## New Files (18 total)

1. `app/core/service_client.py` - HTTP client
2. `app/apideps.py` - API dependencies
3. `app/services/metric_service.py` - Metric service
4. `app/services/dashboard_service.py` - Dashboard service
5. `app/services/data_sync_service.py` - Data sync service
6. `app/services/partner_analytics_service.py` - Partner analytics
7. `app/services/project_analytics_service.py` - Project analytics
8. `app/services/social_media_analytics_service.py` - Social media analytics
9. `app/services/notification_analytics_service.py` - Notification analytics
10. `app/services/aggregation_service.py` - Aggregation orchestration
11. `app/api/v1/endpoints/metrics.py` - Metrics endpoints

12. `app/api/v1/endpoints/dashboards.py` - Dashboard endpoints
13. `app/api/v1/endpoints/data_sync.py` - Data sync endpoints
14. `app/api/v1/endpoints/partner_analytics.py` - Partner analytics endpoints
15. `app/api/v1/endpoints/project_analytics.py` - Project analytics endpoints
16. `app/api/v1/endpoints/social_media_analytics.py` - Social media endpoints
17. `app/api/v1/endpoints/notification_analytics.py` - Notification endpoints
18. `ANALYTICS_SERVICE_API_SUMMARY.md` - This document

## Modified Files (3 total)

1. `app/core/config.py` - Added service URLs and settings
  2. `.env.example` - Added new environment variables
  3. `app/api/v1/api.py` - Registered all new routers
- 

## Deployment Checklist

- [ ] Run database migrations: `alembic upgrade head`
  - [ ] Configure environment variables
  - [ ] Set up Redis for caching
  - [ ] Configure Kafka for events
  - [ ] Set up monitoring (DataDog)
  - [ ] Configure CORS origins
  - [ ] Set `SECRET_KEY` in production
  - [ ] Enable HTTPS
  - [ ] Set up log aggregation
  - [ ] Configure service discovery
  - [ ] Set up health checks
  - [ ] Configure auto-scaling
- 

## Support and Maintenance

### Logs

- Structured logging with correlation IDs
- JSON format for log aggregation
- Configurable log levels

### Monitoring

- Health check endpoint: `/api/v1/health`
- DataDog integration ready
- Metrics and tracing support

### Documentation

- OpenAPI/Swagger specification
- Inline code documentation

- Architecture diagrams (see roadmap docs)
- 

**Implementation Complete:** December 24, 2025

**Status:** Ready for testing and integration

**Next Steps:** Deploy to development environment and begin integration testing