



**ND ESTATES**  
Advanced Analytics & Marketing Intelligence

## NDE-Stats-GA - Architecture & Design Document

---

### ■■■ **IMPORTANT: INTERNAL-USE ONLY**

- \*\*Access Restriction\*\*: Staff members and authorized personnel only
- \*\*Deployment\*\*: Private network with restricted authentication
- \*\*Data Scope\*\*: Internal analytics and marketing metrics
- \*\*Security Level\*\*: Enterprise-grade security for internal operations
- \*\*Backup & Recovery\*\*: Internal systems, no public data handling

---

## Table of Contents

1. [System Overview](#system-overview)
2. [Architecture Layers](#architecture-layers)
3. [Core Components](#core-components)
4. [Technology Stack](#technology-stack)
5. [Data Flow](#data-flow)
6. [API Integration](#api-integration)
7. [Security Architecture](#security-architecture)
8. [Deployment Architecture](#deployment-architecture)
9. [Scalability & Performance](#scalability--performance)
10. [Development Patterns](#development-patterns)
11. [Testing Strategy](#testing-strategy)

---

# System Overview

## Purpose

NDE-Stats-GA is a comprehensive internal analytics and marketing tool that aggregates data from multiple sources:

- Google Analytics 4 (GA4)
- Google Ads API
- Google Search Console (GSC)

## Target Users

- **Internal Staff**: ND Estates data analysts, marketing team, management
- **Administrators**: System operators, developers
- **Not Intended For**: Public users, external partners (unless explicitly granted access)
- **Restricted Access**: Staff members only, requires authentication

## Key Objectives

- Real-time analytics dashboarding for internal decision-making
- Automated report generation (daily, weekly, monthly)
- Multi-channel marketing performance tracking
- Audience segmentation and management
- Trend analysis and insights delivery

---

## Architecture Layers

### 1. **Presentation Layer** (User Interface)

#### Web Dashboard (Primary)

- **Technology**: PHP 5.7+ with Bootstrap 5
- **Location**: `/web/` directory
- **Components**:
  - `index.php` - Main dashboard interface
  - `run\_report.php` - Script execution handler
  - `documentation.php` - In-app documentation
  - `css/` - Styling assets
  - `js/` - Client-side logic (minimal)

#### Alternative Flask Interface (Secondary)

- **Technology**: Python Flask
- **Location**: Root `app.py`
- **Purpose**: Alternative Python-based web interface
- **Status**: Secondary to PHP frontend

## Access Control

- **Authentication**: Session-based with bcrypt password hashing
- **Authorization**: Role-based access (admin/user roles)

- \*\*Timeout\*\*: Automatic session expiration
- \*\*HTTPS Only\*\*: All traffic encrypted in production

## 2. \*\*Application Layer\*\* (Business Logic)

### Analysis Scripts ( `scripts` directory)

20+ specialized Python scripts for specific analytics tasks:

- `hourly\_traffic\_analysis.py` - Hourly traffic patterns
- `page\_traffic\_analysis.py` - Per-page performance
- `yesterday\_report.py` - Daily report generation
- `top\_pages\_report.py` - Best performing pages
- `channel\_report.py` - Channel grouping analysis
- `google\_ads\_performance.py` - Campaign effectiveness
- `create\_ad.py` - Programmatic ad creation
- `audience\_management.py` - GA4 audience operations
- `social\_media\_timing.py` - Optimal posting times
- `keyword\_analysis.py` - Combined GSC/GA4 insights
- `conversion\_funnel\_analysis.py` - User journey tracking

### Core Module Library ( `src` directory)

- Environment variable management
- Credential loading and validation
- Property ID and customer ID definitions
- API key initialization
- GA4 API wrapper and utilities
- Query builder for GA4 reports
- Date range helpers
- Response parsing and validation
- Retry logic for API resilience
- PDF report generation
- Template rendering
- Data visualization in PDFs
- Export formatting

### Application Server

- \*\*Technology\*\*: PHP-FPM (production) or Flask development server
- \*\*Port\*\*: 80/443 (web) or 5000 (Flask)
- \*\*Concurrency\*\*: Handles multiple simultaneous requests
- \*\*Logging\*\*: Centralized application logging

## 3. \*\*Data Layer\*\* (Storage & Caching)

### File-Based Report Storage

- \*\*Location\*\*: `/reports/` directory
- \*\*Format\*\*: CSV files with date-stamped naming
- \*\*Naming Convention\*\*: `{report\_type}\_{start\_date}\_to\_{end\_date}.csv`

- \*\*Retention\*\*: Indefinite (local storage)
- \*\*Purpose\*\*: Archival and historical analysis

## Temporary Data

- \*\*Location\*\*: `/tmp/` (system temp)
- \*\*Purpose\*\*: Intermediate processing
- \*\*Lifecycle\*\*: Cleaned up after each script execution

## Configuration Storage

- \*\*Location\*\*: ` `.env` file (gitignored)
- \*\*Purpose\*\*: Environment-specific credentials and settings
- \*\*Security\*\*: Never committed to version control

## Log Files

- \*\*Location\*\*: `/logs/` directory
- \*\*Types\*\*:
  - `web\_interface.log` - PHP application logs
  - `{script\_name}.log` - Individual script execution logs
  - `rate\_limits.json` - API rate limiting state
- \*\*Rotation\*\*: Manual cleanup (recommended weekly)
- \*\*Retention\*\*: Last 30 days recommended

## Database (Optional)

- \*\*Technology\*\*: MariaDB 10.3+
  - \*\*Purpose\*\*: Optional persistence layer
  - \*\*Status\*\*: Available but not required for core functionality
  - \*\*Use Cases\*\*: Long-term metric storage, user preferences
- 

## Core Components

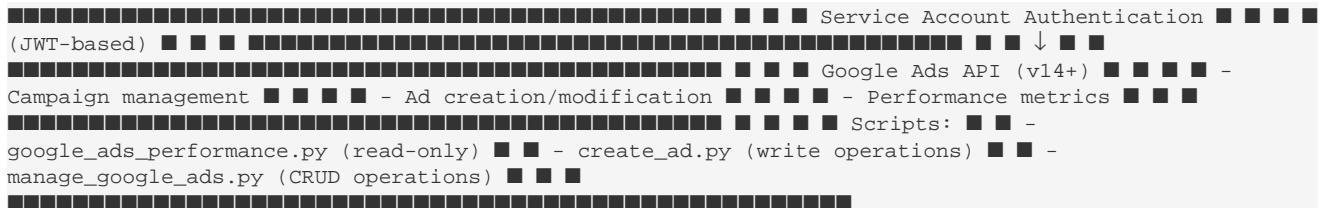
### GA4 Integration Module



- Dimension selection (page, source, medium, campaign, etc.)
- Metric calculation (users, sessions, engagement, conversions)
- Date range filtering (custom ranges)
- Aggregation and rollups
- Real-time data access (up to ~30 minutes latency)

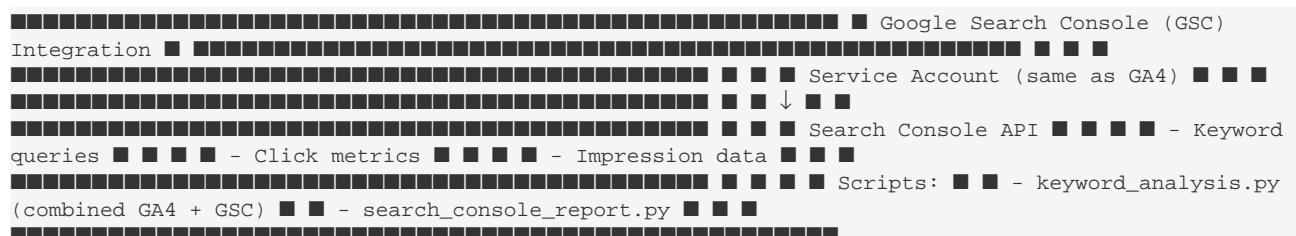
### Google Ads Integration Module





- Email: Defined in Google Cloud Console
- Manager Account ID: Login credentials for MCC access
- Customer Account ID: Target account to manage
- Developer Token: Required for production access
- Permissions: Admin role on manager account

## Search Console Integration Module

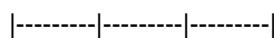


## Technology Stack

### Backend Framework



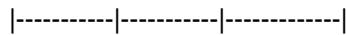
### Python Libraries



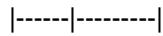
### Frontend Libraries



### Infrastructure



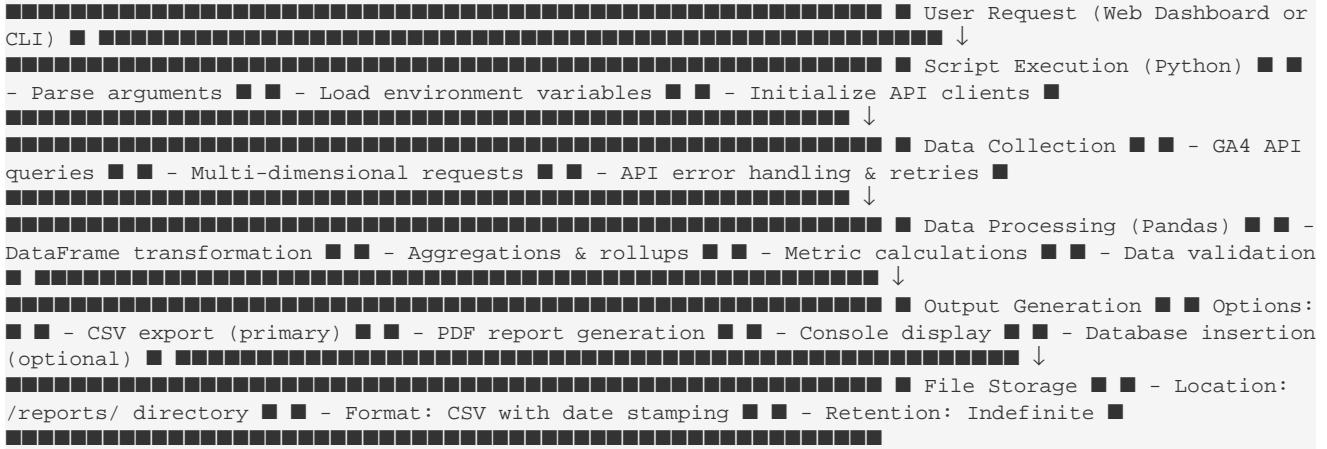
### Security & Compliance



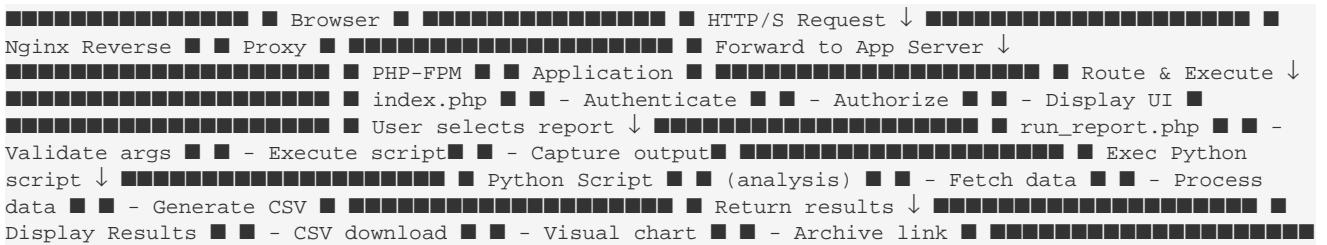
--

## Data Flow

### Report Generation Workflow



## Web Request Flow



## API Integration

### Google Analytics 4 (GA4) API

- Service Account JSON key (JWT-based)
- OAuth 2.0 Refresh Token (OAuth-based)
- Scopes: `https://www.googleapis.com/auth/analytics.readonly`
- 10,000 requests per day (free tier)
- 1,000,000 requests per day (Basic access)
- Concurrent request limit: 10
- Real-time data: ~30 minutes latency
- Intraday data: ~4 hours latency
- Complete data: Next day at ~10:30 AM PT

```

# Dimensions (breakdown attributes)
dimensions = ['date', 'sourceMedium', 'pageTitle', 'eventName'] #
Metrics (measurements)
metrics = ['activeUsers', 'sessions', 'engagementRate', 'conversions'] #
Filters (conditions)
filters = {'stringFilter': {'value': '/product'}} # Date ranges
date_ranges = [{"startDate": 'YYYY-MM-DD', 'endDate': 'YYYY-MM-DD'}]

```

### Google Ads API

- Service Account (for read-only access via MCC structure)
- OAuth 2.0 (for account-level operations)
- Developer Token (required for production access)
- Manager Account (MCC) - Service account added here

- Child Accounts - Managed through manager account
- Target Account - Specific account to operate on
- Daily budget: ~15,000 operations/day
- Hourly limit: ~2,500 operations/hour
- Burst capacity: Limited
- Read campaigns, ad groups, keywords
- Create/update ads and keywords
- Manage audience lists
- Retrieve performance metrics

## Google Search Console API

- Service Account (same as GA4)
  - Scopes: `https://www.googleapis.com/auth/webmasters.readonly`
  - Search queries & performance
  - Click metrics
  - Impressions
  - Average position
  - Crawl statistics
  - Coverage reports
  - 600 requests per minute
  - 200,000 requests per day
- 

## Security Architecture

### Authentication & Access Control

#### Internal-Only Access

- \*\*VPN/Network Restriction\*\*: Access only from internal network
- \*\*IP Whitelisting\*\*: Only authorized staff IPs allowed
- \*\*Firewall Rules\*\*: DigitalOcean firewall configured

#### User Authentication

- \*\*Session-Based\*\*: Cookie-based sessions with secure flags
- \*\*Password Hashing\*\*: bcrypt with configurable rounds
- \*\*Password Requirements\*\*:
  - Minimum 12 characters
  - Mixed case, numbers, special characters
  - No common patterns
- \*\*Account Lockout\*\*: 5 failed attempts → 30-minute lockout
- \*\*Session Timeout\*\*: 1 hour inactivity expiration

#### Role-Based Authorization

- \*\*Admin\*\*: Full system access, user management
- \*\*Analyst\*\*: Report access, limited configuration

- \*\*Viewer\*\*: Read-only access to specific reports
- \*\*Custom Roles\*\*: Define as needed

## Credential Management

### API Keys & Secrets

```
[REDACTED] Environment Variables (.env) [REDACTED]
[REDACTED] GOOGLE_ADS_CUSTOMER_ID [REDACTED] [REDACTED]
GOOGLE_ADS_CLIENT_ID [REDACTED] GOOGLE_ADS_CLIENT_SECRET [REDACTED] GOOGLE_ADS_DEVELOPER_TOKEN [REDACTED] [REDACTED]
GOOGLE_ADS_JSON_KEY_PATH [REDACTED] GA4_PROPERTY_ID [REDACTED] GA4_KEY_PATH [REDACTED] DATABASE_PASSWORD [REDACTED]
[REDACTED] ↓ (File Not in Git)
[REDACTED] Encrypted Storage [REDACTED] (AES-256 Encryption) [REDACTED]
```

### Service Account Keys

- \*\*Location\*\*: `ddev/keys/` (gitignored)
- \*\*Type\*\*: JSON key files from Google Cloud
- \*\*Rotation\*\*: Quarterly (recommended)
- \*\*Backup\*\*: Secure offline backup maintained
- \*\*Revocation\*\*: Immediate if compromised

### OAuth Tokens

- \*\*Refresh Tokens\*\*: Long-lived, securely stored
- \*\*Access Tokens\*\*: Short-lived (1 hour), auto-refreshed
- \*\*Token Rotation\*\*: Automatic on each refresh
- \*\*Revocation\*\*: Can be revoked via Google Dashboard

## Data Protection

### In-Transit

- \*\*HTTPS/TLS 1.3\*\*: All network traffic encrypted
- \*\*Certificate Management\*\*: Auto-renewal via Let's Encrypt
- \*\*HSTS\*\*: HTTP Strict Transport Security enabled
- \*\*Pinning\*\*: Certificate pinning for critical endpoints

### At-Rest

- \*\*CSV Files\*\*: File permissions 0600 (owner read-write only)
- \*\*Database\*\*: Encrypted columns for sensitive data
- \*\*Backups\*\*: Encrypted offsite backups
- \*\*Temporary Files\*\*: Secure cleanup after processing

### API Communication

- \*\*TLS Verification\*\*: All SSL/TLS certs verified
- \*\*Request Signing\*\*: JWT signing for service accounts
- \*\*Response Validation\*\*: Checksum verification
- \*\*Replay Prevention\*\*: Timestamp/nonce validation

## Monitoring & Logging

### Security Logging

/logs/security.log ■■■ Login attempts (success/failure) ■■■ Authorization failures ■■■ API credential access ■■■ Permission changes ■■■ Data export operations ■■■ Suspicious activity

## Audit Trail

- **\*\*What\*\***: Who did what, when, where
  - **\*\*Retention\*\***: 12 months minimum
  - **\*\*Access\*\***: Admin-only review
  - **\*\*Immutability\*\***: Write-once, read-many (WORM)

## Real-Time Alerts

- **Failed Logins**: >3 in 5 minutes
  - **Rate Limit Hits**: API quota exceeded
  - **Permission Errors**: Unauthorized access attempts
  - **Configuration Changes**: Any system setting modification

# Network Security

## Firewall Configuration

Inbound Rules: ■■■ Port 443 (HTTPS) - Allow staff IPs only ■■■ Port 22 (SSH) - Allow admin IPs only ■■■ Port 80 (HTTP) - Redirect to HTTPS only ■■■ All other - DENY Outbound Rules: ■■■ Port 443 (HTTPS) - Allow (Google APIs, etc.) ■■■ Port 5432 (PostgreSQL) - Allow (internal DB only) ■■■ All others - DENY

## DDoS Protection

- **\*\*Rate Limiting\*\***: 100 requests/minute per IP
  - **\*\*Request Size Limit\*\***: 50MB maximum
  - **\*\*Connection Timeout\*\***: 30 seconds idle
  - **\*\*Cloudflare\*\***: Optional CDN/DDoS protection

## Deployment Architecture

## Local Development Environment

## Production Deployment

```
graph TD; A[DigitalOcean App Platform] --> B[Nginx - Reverse Proxy & Load Balancer]; B --> C[TLS Termination (Let's Encrypt)]; B --> D[Rate Limiting]; B --> E[IP Whitelisting]; B --> F[Request Logging]; F --> G[PHP-FPM Application Container(s)]; G --> H[Multiple instances (auto-scaled)]; G --> I[Health checks (HTTP 200 response)]; G --> J[Session storage (Redis or Memcache)]; G --> K[Environment variable injection]; K --> L[Persistent Storage]; L --> M[PostgreSQL]
```

The diagram illustrates the DigitalOcean App Platform architecture. It starts with the "DigitalOcean App Platform" at the top, which connects to an "Nginx - Reverse Proxy & Load Balancer". This proxy handles "TLS Termination (Let's Encrypt)", "Rate Limiting", "IP Whitelisting", and "Request Logging". Below the proxy are "PHP-FPM Application Container(s)". These containers support "Multiple instances (auto-scaled)", "Health checks (HTTP 200 response)", "Session storage (Redis or Memcache)", and "Environment variable injection". Finally, "Persistent Storage" (using "PostgreSQL") is connected to the application containers.

Database █ █ █ █ - Object Storage (for reports) █ █ █ █ - Encrypted Backups █ █ █  
███████████████████ ██████████ █████ █ █

- Let's Encrypt certificates
- Auto-renewal 30 days before expiry
- HSTS headers enabled
- TLS 1.2+ minimum

## CI/CD Pipeline

```
Git Push ↓ [1] Run Tests ███ Unit tests ███ Integration tests ███ Code quality checks ↓ [2] Build Artifacts ███ Docker image creation ███ Asset compilation ███ Version tagging ↓ [3] Security Scan ███ Dependency vulnerabilities ███ Secret detection ███ Container image scan ↓ [4] Stage Deployment (Staging) ███ Deploy to staging environment ███ Run smoke tests ███ Performance benchmarks ↓ [5] Production Deployment (Manual) ███ Manual approval required ███ Blue-green deployment ███ Health checks ███ Rollback capability
```

## Backup & Disaster Recovery

- **Frequency**: Daily automated backups
- **Retention**: 30-day rolling window
- **Location**: Encrypted offsite storage
- **Verification**: Weekly restore tests

```
1. Detect failure (automated alerts) 2. Assess impact (manual review) 3. Restore from backup (automated script) 4. Verify data integrity (automated tests) 5. Resume operations (manual activation) 6. Post-mortem analysis (documented)
```

- **RTO** (Recovery Time Objective): 4 hours
- **RPO** (Recovery Point Objective): 24 hours

## Scalability & Performance

### Horizontal Scaling

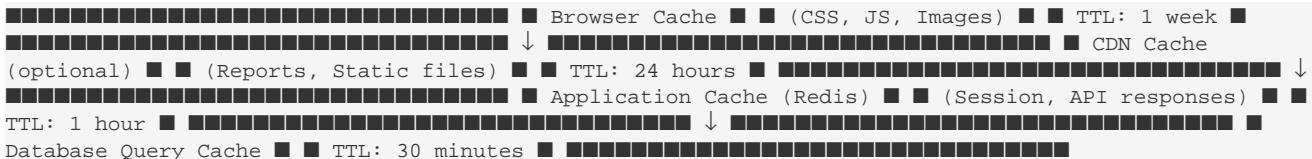
- Nginx as reverse proxy
- Round-robin distribution
- Session persistence (sticky sessions)
- Health check monitoring

```
CPU Usage: ███ < 30% - Scale down (remove instances) ███ 30-70% - Maintain current ███ > 70% - Scale up (add instances) Memory Usage: ███ < 40% - Scale down ███ 40-80% - Maintain current ███ > 80% - Scale up Max Instances: 5 Min Instances: 2
```

### Vertical Scaling

- **PHP-FPM**: 2 CPU, 2GB RAM per container
- **Database**: 4 CPU, 8GB RAM
- **Total**: Scalable based on demand

## Caching Strategy



## Performance Optimization

- Batch requests when possible
  - Use appropriate time granularity
  - Limit dimensions per query
  - Implement exponential backoff for retries
  - Stream processing for large datasets
  - Chunked DataFrame operations
  - Lazy evaluation where possible
  - Memory-efficient CSV writing
  - Gzip compression (enabled)
  - CSS/JS minification
  - Image optimization
  - Lazy loading for tables
- 

## Development Patterns

### Standard Script Structure

```

#!/usr/bin/env python3 """ Script description and purpose. """
import sys
import os
import datetime
# Add parent directory to path for imports
sys.path.insert(0, os.path.join(os.path.dirname(__file__), '..'))
from src.config import GA4_PROPERTY_ID, REPORTS_DIR
from src.ga4_client import run_report, create_date_range, get_yesterday_date
def main():
    """Main execution function."""
    try:
        # 1. Set up date ranges
        start_date = get_yesterday_date()
        end_date = get_yesterday_date()
        date_range = create_date_range(start_date, end_date)
        # 2. Query GA4 API
        response = run_report(dimensions=['date', 'sourceMedium'], metrics=['activeUsers', 'sessions'],
                               date_ranges=[date_range])
        # 3. Validate response if response.row_count == 0:
        if response.row_count == 0:
            print("No data found for the specified date range.")
            return
        # 4. Process with pandas
        import pandas as pd
        data = []
        for row in response.rows:
            data.append({
                'date': row.dimension_values[0].value,
                'source_medium': row.dimension_values[1].value,
                'users': int(row.metric_values[0].value),
                'sessions': int(row.metric_values[1].value)
            })
        df = pd.DataFrame(data)
        # 5. Save to CSV
        filename = f"report_{start_date}_to_{end_date}.csv"
        filepath = os.path.join(REPORTS_DIR, filename)
        df.to_csv(filepath, index=False)
        print(f"Report saved: {filepath}")
    except Exception as e:
        print(f"Error: {e}")
        import traceback
        traceback.print_exc()
    return 1
if __name__ == "__main__":
    sys.exit(main())

```

### Configuration Pattern

```

# src/config.py
import os
from dotenv import load_dotenv # Load environment variables
load_dotenv() # GA4 Configuration
GA4_PROPERTY_ID = os.getenv('GA4_PROPERTY_ID')
GA4_KEY_PATH = os.getenv('GA4_KEY_PATH') # Google Ads Configuration
GOOGLE_ADS_CUSTOMER_ID = os.getenv('GOOGLE_ADS_CUSTOMER_ID')
GOOGLE_ADS_LOGIN_CUSTOMER_ID = os.getenv('GOOGLE_ADS_LOGIN_CUSTOMER_ID')
GOOGLE_ADS_DEVELOPER_TOKEN = os.getenv('GOOGLE_ADS_DEVELOPER_TOKEN')
GOOGLE_ADS_JSON_KEY_PATH = os.getenv('GOOGLE_ADS_JSON_KEY_PATH') # Application Configuration
REPORTS_DIR = os.path.join(os.path.dirname(__file__), '..', 'reports')
LOGS_DIR = os.path.join(os.path.dirname(__file__), '..', 'logs') # Validate critical configuration if not
GA4_PROPERTY_ID or not GA4_KEY_PATH:
    raise ValueError("GA4_PROPERTY_ID and GA4_KEY_PATH must be set in .env")
if not os.path.exists(GA4_KEY_PATH):
    raise FileNotFoundError(f"GA4 key file not found: {GA4_KEY_PATH}")

```

## Error Handling Pattern

```
from google.api_core.exceptions import ( GoogleAPIError, ServiceUnavailable, TooManyRequests ) import time
def run_report_with_retry(request, max_retries=3): """Run report with exponential backoff
retry."""
    for attempt in range(max_retries):
        try:
            return client.get_service(...).search(request=request)
        except (ServiceUnavailable, TooManyRequests) as e:
            if attempt < max_retries - 1:
                wait_time = 2 ** attempt # Exponential backoff
                print(f"Rate limited. Retrying in {wait_time}s...")
                time.sleep(wait_time)
            else:
                raise
        except GoogleAPIError as e:
            print(f"API Error: {e.message}")
            raise
```

## Testing Pattern

```
# tests/test_scripts/test_hourly_traffic.py import unittest from unittest.mock import patch, MagicMock
import sys
import os
sys.path.insert(0, os.path.join(os.path.dirname(__file__), '..', '..'))
from scripts.hourly_traffic_analysis import main
class TestHourlyTrafficAnalysis(unittest.TestCase):
    @patch('src.ga4_client.run_report')
    def test_no_data_handling(self, mock_run_report):
        """Test handling when no data is returned."""
        mock_response = MagicMock()
        mock_response.row_count = 0
        mock_run_report.return_value = mock_response
        result = main()
        self.assertEqual(result, 0) # Should handle gracefully
    @patch('src.ga4_client.run_report')
    def test_successful_report_generation(self, mock_run_report):
        """Test successful report generation."""
        # Setup mock response
        mock_response = MagicMock()
        mock_response.row_count = 1
        ... setup mock data
        mock_run_report.return_value = mock_response
        result = main()
        self.assertEqual(result, 0)
        if __name__ == '__main__':
            unittest.main()
```

--

## Testing Strategy

### Unit Testing

- \*\*Framework\*\*: unittest (Python standard library)
- \*\*Coverage Target\*\*: 80%+ code coverage
- \*\*Scope\*\*: Individual functions, helper utilities
- \*\*Mocking\*\*: API responses, file I/O, external services

### Integration Testing

- \*\*Scope\*\*: Multi-component workflows (script → API → storage)
- \*\*Data\*\*: Test datasets with known outputs
- \*\*Environment\*\*: Isolated test environment with test API credentials
- \*\*Cleanup\*\*: Automatic cleanup after tests

### End-to-End Testing

- \*\*Scope\*\*: Complete user workflows (web UI → script → report)
- \*\*Test Cases\*\*:
  - Report generation flow
  - Authentication & authorization
  - Data export & download
  - Error handling
- \*\*Frequency\*\*: Before each production release

### Performance Testing

- \*\*Load Testing\*\*: Simulate 10+ concurrent users
- \*\*Stress Testing\*\*: Determine breaking point

- \*\*Endurance Testing\*\*: 24-hour sustained load
- \*\*Tool\*\*: Apache JMeter or k6

## Security Testing

- \*\*Static Analysis\*\*: SAST (CodeQL, Semgrep)
- \*\*Dependency Scanning\*\*: Dependabot, Snyk
- \*\*Penetration Testing\*\*: Quarterly by external firm
- \*\*OWASP Top 10\*\*: Verified coverage

---

## Component Dependencies

### Critical Path

```
Application Startup ■■■ Environment variables loaded (.env) ■■■ Google Cloud credentials validated ■■■ Service account keys verified ■■■ Database connection established (if used) ■■■ Cache layer initialized (Redis) ■■■ Script Execution ■■■ API client initialized ■■■ Authentication completed ■■■ Data query constructed ■■■ API request sent ■■■ Response validated ■■■ Data processing ■■■ Output written to storage
```

## Dependency Management

- Specified in `requirements.txt`
- Pinned to minor version (e.g., `2.3.\*`)
- Security patches auto-applied via Dependabot
- Monthly vulnerability scanning
- PHP 7.4+ or 8.1+
- Python 3.11+
- PostgreSQL 12+ (optional)
- Redis 6+ (optional)

---

## Monitoring & Observability

### Metrics Collection

- Request count & latency
- Error rates (5xx, 4xx)
- API quota usage
- Database query performance
- Cache hit/miss rates
- Report generation success rate
- Data freshness (last update time)
- User engagement (logins, reports accessed)
- Data quality (missing data %)

### Health Checks

```
{ "status": "ok", "timestamp": "2026-01-03T10:30:00Z", "checks": { "database": "ok", "ga4_api": "ok", "google_ads_api": "ok", "storage": "ok", "cache": "ok" } }
```

## Alerting

- Application down
- Database connection lost
- Authentication failure
- API quota exceeded
- High error rates (>5%)
- Slow response times (>5s)
- Cache failure
- Low disk space (<10%)

---

## Maintenance & Operations

### Scheduled Maintenance

- **Backup verification**: Daily
- **Security updates**: Weekly
- **Database optimization**: Weekly
- **Log rotation**: Daily
- **Dependency updates**: Monthly

### Runbook Examples

1. Stop application (graceful shutdown)
  2. Backup current database
  3. Restore from latest backup
  4. Verify data integrity
  5. Resume application
  6. Monitor error logs
  7. Document incident
- 
1. Monitor CPU/Memory metrics
  2. Trigger manual scaling (if auto-scaling fails)
  3. Add new container instances
  4. Update load balancer configuration
  5. Run health checks
  6. Monitor for 30 minutes

---

## Glossary

|-----|-----|

---

## Conclusion

NDE-Stats-GA is a secure, scalable internal analytics system designed specifically for ND Estates staff use. With robust API integrations, comprehensive security controls, and a modern deployment architecture, it provides the foundation for data-driven decision-making within the organization.

All access is restricted to authorized internal staff only, with enterprise-grade security measures in place to protect sensitive analytics and marketing data.