Anwar Sales Ecosystem - Unified Google Apps Script

A comprehensive, unified Google Apps Script project that consolidates all sales ecosystem functionality into a single, well-structured codebase.

🚀 Project Overview

This project unifies the legacy `appscriptnew` and `unified-apps-script` projects into a single, modern, and maintainable Google Apps Script project following best practices for directory structure and code organization.

📁 Directory Structure

```
src/
├── appsscript.json          # Apps Script configuration
├── config/
│   └── Config.js            # Centralized configuration
├── database/
│   └── DatabaseService.js   # Unified database operations
├── services/
│   └── WhatsAppService.js   # WhatsApp messaging service
├── handlers/
│   ├── MainHandler.js       # Main entry points and routing
│   ├── EngineerHandler.js   # Engineer registration handling
│   ├── PotentialSiteHandler.js  # Potential site registration handling
│   ├── RetailerHandler.js   # Retailer registration handling
│   └── CRMHandler.js        # General CRM operations
├── legacy/
│   ├── InitializeSheets.js  # Legacy sheet initialization
│   ├── SetupTriggers.js     # Legacy trigger setup
│   └── config/              # Legacy configuration (if needed)
├── utilities/               # Shared utility functions
└── sheets/                  # Legacy sheet operations
```

🔄 Migration Summary

Legacy Features Preserved

- **Sheet Initialization**: `InitializeSheets.js` maintains backward compatibility
- **Form Data Import**: Legacy form data import functionality
- **Trigger Setup**: Daily automation triggers
- **Configuration**: All legacy configuration values preserved

New Features Added

- **Unified Database Service**: Centralized Google Sheets operations

- **Improved WhatsApp Integration**: Enhanced messaging with error handling
- **Modular Architecture**: Clean separation of concerns
- **Better Error Handling**: Comprehensive logging and error management
- **Scalable Structure**: Easy to extend and maintain

🛠️ Setup Instructions

1. Initial Setup

```
# From the project root directory
cd E:\Anwar_sales_eco

# Login to Google Apps Script
npx @google/clasp login

# Create new Apps Script project (optional)
npx @google/clasp create --title "Anwar Sales Ecosystem" --type sheets
```

2. Configuration

Update `src/config/Config.js` with your actual:

- Google Form IDs
- Spreadsheet IDs
- Maytapi API credentials
- Any custom configuration values

3. Deployment

```
# Push code to Apps Script
npx @google/clasp push

# Open in browser
npx @google/clasp open
```

4. Initial Setup Function

Run the `setup()` function from `MainHandler.js` to:

- Initialize all required sheets
- Set up automated triggers
- Configure the environment

📋 Available Functions

Entry Points

- `doGet(e)` - Web endpoint for form routing
- `onFormSubmit(e)` - Form submission handler
- `onEdit(e)` - Sheet edit handler
- `setup()` - Initial setup function

Legacy Functions

- `initializeNewSheets()` - Legacy sheet initialization
- `setupTriggers()` - Legacy trigger setup

🔧 Configuration

Required Properties

Set these in your Google Apps Script project properties:

- `MAYTAPI_API_KEY` - Your Maytapi API key
- `MAIN_SPREADSHEET_ID` - Your main CRM spreadsheet ID

Form IDs (Update these)

- `ENGINEER_FORM_ID` - Engineer registration form
- `POTENTIAL_SITE_FORM_ID` - Potential site registration form
- `RETAILER_FORM_ID` - Retailer registration form

🚀 Usage Examples

Adding a New Registration Type

1. Add new configuration in `config/Config.js`
2. Create a new handler in `handlers/`
3. Update routing in `handlers/MainHandler.js`

Custom WhatsApp Messages

```
// Use the WhatsAppService for custom messages
WhatsAppService.sendMessage(phone, message);
```

Database Operations

```
// Insert a new record
DB.insertRecord('SheetName', data);

// Get all records
var records = DB.getAllRecords('SheetName');
```

```
// Find specific records
var results = DB.findRecords('SheetName', {Status: 'Pending'});
```

Best Practices

1. **Use the unified Config.js** for all configuration
2. **Leverage DatabaseService** for all sheet operations
3. **Implement proper error handling** in all handlers
4. **Use consistent naming conventions** across files
5. **Add console logging** for debugging and monitoring

Troubleshooting

Common Issues

- **Sheet not found**: Ensure sheets are initialized via `setup()`
- **API key issues**: Verify Maytapi API key in project properties
- **Form ID errors**: Check form IDs in configuration

Debug Mode

Enable detailed logging by setting:

```
console.log('Debug: Your message here');
```

Support

For issues or questions:

1. Check the Apps Script execution logs
2. Verify all configuration values
3. Ensure all required sheets exist
4. Test individual handlers independently

Future Enhancements

- [ ] Add more comprehensive error handling
- [ ] Implement data validation
- [ ] Add reporting capabilities
- [ ] Create admin dashboard
- [ ] Add batch processing features
- [ ] Implement user management
- [ ] Add analytics and insights

Detailed Product Requirements Document (PRD)

Overview

The Anwar Sales Ecosystem is a unified Google Apps Script project designed to streamline sales operations, data management, and communication. It integrates legacy systems with modern functionalities, providing a scalable and maintainable solution.

Modules

1. Configuration

- **Purpose**: Centralized management of constants and settings.
- **Key Features**:
    - Sheet names and headers.
    - Form IDs for Google Forms.
    - API keys and URLs.

2. Database Service

- **Purpose**: Abstracts Google Sheets operations.
- **Key Features**:
    - Sheet initialization and header management.
    - Record insertion, updating, and retrieval.
    - Data validation and ID generation.

3. Handlers

- **Purpose**: Manage form submissions and sheet edits.
- **Key Features**:
    - Routing based on form IDs.
    - Data validation and processing.
    - Integration with WhatsApp for notifications.

4. Services

- **Purpose**: Provide specialized functionalities.
- **Key Features**:
    - WhatsApp messaging.
    - Location-based operations.

- Test environment setup.
- Data migration.

5. Legacy Support

- **Purpose**: Ensure backward compatibility.
- **Key Features**:
  - Legacy sheet initialization.
  - Legacy trigger setup.

Functional Requirements

1. Form Submission

- **Input**: Data from Google Forms.
- **Process**:
  - Validate data.
  - Insert into appropriate sheets.
  - Notify relevant users.
- **Output**: Updated sheets and notifications.

2. Sheet Edit

- **Input**: Edited data in Google Sheets.
- **Process**:
  - Validate changes.
  - Update related records.
  - Notify relevant users.
- **Output**: Updated records and notifications.

3. WhatsApp Integration

- **Input**: Message templates and recipient data.
- **Process**:
  - Format messages.
  - Send via Maytapi API.

- **Output**: Sent messages and logs.

4. Data Migration

- **Input**: Legacy form data.
- **Process**:
    - Map data to new structure.
    - Insert into sheets.
- **Output**: Migrated data.

Non-Functional Requirements

- **Scalability**: Support for additional modules and forms.
- **Maintainability**: Modular architecture.
- **Reliability**: Error handling and logging.

Future Enhancements

- **Reporting**: Add analytics and insights.
- **Dashboard**: Create an admin interface.
- **Batch Processing**: Implement bulk operations.

This PRD serves as a comprehensive guide for understanding and extending the Anwar Sales Ecosystem.