EET Shopify Sync Application

A Node.js application that automatically syncs products from EET pricing files to Shopify stores. The application filters products based on configurable rules and handles product creation, updates, and inventory management.

🚀 Features

- **CSV Product Parsing**: Reads EET pricing files with semicolon delimiters
- **Smart Filtering**: Configurable include/exclude rules for brands and SKUs
- **Shopify Integration**: Full GraphQL API integration for product management
- **Product Mapping**: Converts EET product data to Shopify format
- **Inventory Sync**: Updates prices and stock quantities
- **Draft Management**: Makes orphaned products draft automatically
- **Scheduled Execution**: Runs every 12 hours or on-demand
- **Comprehensive Logging**: Detailed logs with timestamps and rotation
- **EET API Integration**: Real-time price and stock updates

📋 Prerequisites

- Node.js 18+
- Shopify store with Admin API access
- EET API credentials (for price/stock updates)
- EET pricing file (`eet_prices.txt`)

🛠️ Installation

1. **Clone the repository**

```
git clone <repository-url>
cd eet-shopify
```

2. **Install dependencies**

```
npm install
```

3. **Set up environment variables**
   Create a `.env` file in the root directory:

```
# Environment Configuration
PRODUCTION=development
LANGUAGE=DNK
```

```
EET_PRICE=eet_prices.txt

# Shopify Configuration (Development)
SHOPIFY_TEST_STORE_ADMIN_URL=fetch-products-from-eet.myshopify.com
SHOPIFY_TEST_STORE_ADMIN_API=shpat_your_access_token_here

# Shopify Configuration (Production)
SHOPIFY_PRODUCTION_STORE_ADMIN_URL=your-store.myshopify.com
SHOPIFY_PRODUCTION_STORE_ADMIN_API=shpat_your_production_token_here

# Scheduler Configuration
SCHEDULED_MODE=false

# Logging Configuration
LOGGING=true
```

4. **Place your EET pricing file**
    - Name it `eet_prices.txt` (or update `EET_PRICE` in `.env`)
    - Ensure it's in the root directory

🎯 Usage

Single Run Mode (Default)

```
npm start
```

Scheduled Mode (Every 12 Hours)

```
npm run start:scheduled
```

🔧 Configuration

Product Filtering

The application uses `config/product-filter.json` to define which products to sync:

```
{
  "include": {
    "brand": [
      "Axis",
      "Hikvision",
      "Sony"
    ],
    "sku": [
```

```
      "PRODPAM10",
      "SPECIAL001"
    ]
  },
  "exclude": {
    "brand": [
      "Samsung",
      "LG"
    ],
    "sku": [
      "Laptops",
      "Smartphones"
    ]
  },
  "include_products_limit": 50
}
```

Filter Rules

Include Rules (OR logic):

- If a product matches ANY brand in `include.brand` OR ANY SKU in `include.sku`, it will be included
- If both `include.brand` and `include.sku` are empty, all products are considered for inclusion

Exclude Rules (AND logic):

- If a product matches a brand in `exclude.brand` AND a SKU in `exclude.sku`, it will be excluded
- Products matching only one exclude criteria will still be included

Product Limit:

- `include_products_limit`: Maximum number of products to process (0 = no limit)

Example Filter Scenarios

Scenario 1: Include only specific brands

```
{
  "include": {
    "brand": ["Axis", "Hikvision"]
  },
  "exclude": {},
  "include_products_limit": 100
}
```

Result: Only products from Axis or Hikvision brands, up to 100 products.

Scenario 2: Exclude specific products

```
{
  "include": {},
  "exclude": {
    "sku": ["Laptops", "Phones"]
  },
  "include_products_limit": 0
}
```

Result: All products except those with SKUs "Laptops" or "Phones".

Scenario 3: Complex filtering

```
{
  "include": {
    "brand": ["Axis", "Hikvision"]
  },
  "exclude": {
    "sku": ["OLD001", "TEST002"]
  },
  "include_products_limit": 50
}
```

Result: Axis or Hikvision products, excluding specific SKUs, limited to 50 products.

EET Pricing File Format

The application expects a CSV file with semicolon (`;`) delimiters. Required columns:

Column	Description	Example
`Varenr.`	Product SKU	`PROD001`
`Beskrivelse`	Product title	`Security Camera`
`Pris`	Price	`1250.50`
`Lagerbeholdning`	Stock quantity	`25`
`Mærke Navn`	Brand name	`Axis`
`Web Category Name`	Category	`Security`
`Web Picture URL`	Image URL	`https://...`
`Beskrivelse 2`	Description 2	`Additional info`
`Beskrivelse 3`	Description 3	`More details`
`EAN/UPC`	Barcode	`1234567890123`
`Bruttovægt`	Weight (kg)	`1.5`

Column	Description	Example
`Manufacturer Part No`	MPN	`MPN123456`

🔄 Application Flow

1. **Load Configuration**: Reads environment variables and filter config
2. **Fetch Shopify Products**: Gets all existing products from Shopify
3. **Parse EET File**: Reads and parses the EET pricing CSV
4. **Apply Filters**: Filters products based on configuration rules
5. **Map Products**: Converts EET format to Shopify product format
6. **Compare Products**: Identifies new, existing, and orphaned products
7. **Create Products**: Uploads new products to Shopify
8. **Update Prices**: Updates prices from EET API
9. **Update Inventory**: Updates stock quantities
10. **Manage Drafts**: Makes orphaned products draft
11. **Log Results**: Records all operations with timestamps

📊 Logging

Log Files

- **Location**: `logs/` directory
- **Format**: `eet-shopify-YYYY-MM-DD-HH-MM-SS.log`
- **Rotation**: Daily at 00:00
- **Retention**: Configurable (default: 30 days)

Log Levels

- **INFO**: General application flow
- **DEBUG**: Detailed operation information
- **ERROR**: Error conditions and failures
- **WARN**: Warning conditions

Sample Log Entry

```
[2024-01-15T10:30:45.123Z] [INFO] [SHOPIFY] Product created successfully
{
  "productId": "gid://shopify/Product/123456789",
  "title": "Security Camera",
  "sku": "PROD001",
  "vendor": "Axis"
}
```

🚨 Error Handling

The application includes comprehensive error handling:

- **Network Errors**: Retries and graceful degradation
- **API Rate Limits**: Automatic delays between requests
- **Validation Errors**: Detailed error messages
- **File System Errors**: Proper error logging and recovery
- **GraphQL Errors**: Parsed and logged error details

🔧 Troubleshooting

Common Issues

1. "Cannot find package" errors

```
npm install
```

2. "Missing Shopify configuration" error

- Check your `.env` file
- Verify `SHOPIFY_TEST_STORE_ADMIN_URL` and `SHOPIFY_TEST_STORE_ADMIN_API`

3. "EET pricing file not found"

- Ensure `eet_prices.txt` exists in root directory
- Check `EET_PRICE` environment variable

4. "GraphQL errors"

- Verify Shopify API token permissions
- Check API version compatibility
- Review rate limiting

5. Filter not working as expected

- Check `config/product-filter.json` syntax
- Verify column names match EET file headers
- Test with `include_products_limit: 1` for debugging

Debug Mode

Enable detailed logging:

```
LOGGING=true
```

Check logs in `logs/` directory for detailed operation information.

📁 Project Structure

```
```

```
eet-shopify/
├── config/
│   └── product-filter.json      # Filter configuration
├── logs/                        # Log files directory
├── module/
│   ├── csvParseAndFilter.js     # CSV parsing and filtering
│   ├── logger.js                # Logging system
│   ├── shopify.js               # Shopify API client
│   └── eet.js                   # EET API client
├── tmp_data/
│   └── filtered-products.json   # Filtered products output
├── .env                         # Environment variables
├── eet_prices.txt              # EET pricing file
├── index.js                     # Main application
├── package.json                 # Dependencies and scripts
└── README.md                    # This file
```

🤝 Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Test thoroughly
5. Submit a pull request

📄 License

ISC License - see LICENSE file for details.

🆘 Support

For issues and questions:

1. Check the troubleshooting section
2. Review log files for error details
3. Create an issue with:
    - Error messages
    - Log file excerpts
    - Configuration details
    - Steps to reproduce

Happy Syncing! 🛒✨