Project Structure Documentation

Overview

This document provides a comprehensive overview of the ProtonDrive Sync project structure. The structure is designed to be clean, maintainable, and follows best practices for Python application development.

Note: This structure matches our FlexibleDietMenu project to maintain consistency across our repositories.

Directory Layout

```
protondrive-sync/
├── protondrive_sync/          # Main application package
├── docs/                      # Documentation
├── Root-level files           # Configuration and scripts
└── .git/                      # Git repository data
```

Detailed Structure

Root Level Files

The root directory contains essential configuration files and scripts:

File	Purpose
`README.md`	Main project overview and documentation
`LICENSE`	MIT License
`.gitignore`	Git ignore patterns
`.abacus.donotdelete`	Marker file for Abacus.AI integration
`requirements.txt`	Python package dependencies
`setup.py`	Python package installation configuration
`run.sh`	Application launcher script
`install.sh`	System-wide installation script
`uninstall.sh`	Uninstallation script
`verify_setup.sh`	Setup verification script

Script Descriptions

`run.sh` - Quick launcher
- Validates Python and rclone installation
- Checks dependencies
- Launches the application
- Usage: `./run.sh`

`install.sh` - System installer
- Detects Linux distribution
- Installs system dependencies
- Sets up Python environment
- Creates desktop integration
- Configures autostart (optional)
- CachyOS-optimized

`uninstall.sh` - Clean removal
- Removes application files
- Optionally preserves user data
- Removes desktop entries
- Cleans up system integration

`verify_setup.sh` - Setup validation
- Checks Python version
- Verifies rclone installation
- Tests rclone configuration
- Validates dependencies
- Checks file permissions

Application Package: `protondrive_sync/`

The main application code resides in the `protondrive_sync/` package (renamed from `src/` for better naming conventions).

```
protondrive_sync/
├── __init__.py          # Package initialization
├── main.py              # Application entry point
├── config_manager.py    # Configuration management
├── rclone_manager.py    # Rclone integration
├── sync_engine.py       # Sync logic and threading
├── gui.py               # GUI implementation
├── tray.py              # System tray integration
└── utils.py             # Utility functions
```

Module Descriptions

`main.py` - Entry Point
- Application initialization
- PyQt5 application setup
- Signal handling
- Main event loop
- Exports: `main()` function

`config_manager.py` - Configuration
- Reads/writes configuration

- Manages selective sync settings
- Validates configuration
- Default configuration handling
- Exports: `ConfigManager` class

`rclone_manager.py` - Rclone Interface
- Executes rclone commands
- Lists ProtonDrive folders
- Estimates sync size
- Launches rclone configuration
- Detects ProtonDrive remote
- Exports: `RcloneManager` class

`sync_engine.py` - Sync Logic
- Manages sync operations
- Threaded sync execution
- Progress monitoring
- Pause/resume functionality
- Safety checks (dry run, size warnings)
- Exports: `SyncEngine` class

`gui.py` - User Interface
- Main application window
- Setup wizard
- Selective sync interface
- Settings panel
- Activity log display
- Exports: `MainWindow`, `SetupWizard` classes

`tray.py` - System Tray
- System tray icon
- Context menu
- Quick sync action
- Status notifications
- Exports: `SystemTray` class

`utils.py` - Utilities
- Logging setup
- Helper functions
- Common utilities
- Exports: Various utility functions

Import Structure

All modules use relative imports within the package:

```
# In main.py
from .config_manager import ConfigManager
from .rclone_manager import RcloneManager
from .sync_engine import SyncEngine
from .gui import MainWindow, SetupWizard
from .tray import SystemTray
from .utils import setup_logging
```

This allows the package to be renamed or moved without breaking imports.

Documentation: `docs/`

All documentation is organized in the `docs/` directory by category:

```
docs/
├── STRUCTURE.md             # This file
├── setup/                   # Setup and installation
│   ├── INSTALLATION_GUIDE.md
│   ├── INSTALLATION_GUIDE.pdf
│   ├── QUICK_START.md
│   ├── QUICK_START.pdf
│   ├── QUICKSTART.md
│   ├── QUICKSTART.pdf
│   ├── PROTONDRIVE_SETUP.md
│   └── PROTONDRIVE_SETUP.pdf
├── features/                # Feature documentation
│   ├── ENHANCEMENT_SUMMARY.md
│   ├── ENHANCEMENT_SUMMARY.pdf
│   ├── selective-sync.md
│   ├── selective-sync.pdf
│   ├── authentication.md
│   └── authentication.pdf
└── deployment/              # Deployment & packaging
    ├── cachyos-optimization.md
    ├── cachyos-optimization.pdf
    ├── packaging.md
    └── packaging.pdf
```

Documentation Categories

`docs/setup/` - Getting Started
- Installation instructions for various distros
- Quick start guide
- ProtonDrive configuration
- First-time setup
- Troubleshooting common setup issues

`docs/features/` - Feature Documentation
- Enhancement summaries
- Selective sync usage
- Authentication methods
- Feature-specific guides

`docs/deployment/` - Advanced Topics
- CachyOS-specific optimizations
- Packaging for various formats (AUR, deb, RPM, Flatpak, etc.)
- Distribution-specific notes
- Performance tuning

Documentation Format

All documentation files are provided in two formats:
- **Markdown (.md)** - For viewing on GitHub and text editors
- **PDF (.pdf)** - For offline reading and printing

PDFs are generated automatically from Markdown using pandoc.

File Organization Principles

1. Separation of Concerns

- **Application code** → `protondrive_sync/`
- **Documentation** → `docs/`
- **Scripts** → Root level
- **Configuration** → Root level

2. Category-Based Documentation

Documentation is organized by purpose:
- Setup guides go in `docs/setup/`
- Feature docs go in `docs/features/`
- Deployment info goes in `docs/deployment/`

3. Clear Entry Points

- **For users:** `README.md` → Quick guides in `docs/setup/`
- **For developers:** `README.md` → `docs/STRUCTURE.md` → Source code
- **For packagers:** `docs/deployment/packaging.md`

4. Self-Contained Scripts

Each shell script is:
- Executable and self-contained
- Well-commented
- Includes error handling
- Provides user-friendly output

Configuration Files

User Configuration

Location: `~/.config/protondrive-sync/config.json`

```
{
  "rclone_remote": "protondrive",
  "local_folder": "/home/user/ProtonDrive",
  "auto_sync_enabled": true,
  "sync_interval_minutes": 30,
  "sync_mode": "full",
  "included_folders": [],
  "excluded_folders": [],
  "bandwidth_limit_kbps": 0
}
```

Rclone Configuration

Location: `~/.config/rclone/rclone.conf`

Contains rclone remote configurations (managed by rclone).

Desktop Integration

Location: `~/.local/share/applications/protondrive-sync.desktop`

Desktop entry file for application menu integration (created by installer).

Development Workflow

Package Naming

The main package is named `protondrive_sync` (with underscore):
- Follows Python package naming conventions
- Avoids hyphen issues in imports
- Clear and descriptive

The repository is named `protondrive-sync` (with hyphen):
- Follows GitHub naming conventions
- User-friendly URL
- Matches project branding

Import Guidelines

Within the package:

```
from .other_module import SomeClass  # Relative import
```

From outside:

```
from protondrive_sync.main import main  # Absolute import
python3 -m protondrive_sync.main        # Module execution
```

Adding New Documentation

When adding documentation:

1. **Determine category:**
   - Setup/installation? → `docs/setup/`
   - Feature explanation? → `docs/features/`
   - Deployment/packaging? → `docs/deployment/`

2. **Create Markdown file:**
   `bash`
   `vim docs/features/new-feature.md`

3. **Generate PDF:**
   `bash`
   `cd docs/features`
   `pandoc new-feature.md -o new-feature.pdf`

4. **Update references:**
   - Add link in `README.md` if appropriate
   - Update this file (`STRUCTURE.md`)
   - Cross-reference from related docs

Adding New Modules

When adding a new Python module:

1. **Create file:**
   `bash`
   `vim protondrive_sync/new_module.py`
2. **Use relative imports:**
   `python`
   `from .config_manager import ConfigManager`
3. **Export classes/functions:**
   `python`
   `__all__ = ['NewClass', 'new_function']`
4. **Update documentation:**
   - Document in module docstring
   - Update `STRUCTURE.md`
   - Add to README if user-facing

File Types and Patterns

Python Files (`.py`)

- Location: `protondrive_sync/`
- Purpose: Application logic
- Convention: snake_case naming

Shell Scripts (`.sh`)

- Location: Root directory
- Purpose: Installation, launching, verification
- Convention: Descriptive names, executable

Markdown Files (`.md`)

- Location: `docs/` subdirectories
- Purpose: Human-readable documentation
- Convention: UPPERCASE for main docs, lowercase for feature docs

PDF Files (`.pdf`)

- Location: Same as corresponding `.md` file
- Purpose: Offline/printable documentation
- Convention: Same name as `.md` file

Maintenance Guidelines

Keeping Structure Clean

1. **No files in root except essentials**
   - Scripts, README, LICENSE, config files only
   - Move other files to appropriate subdirectories

2. **Documentation stays in docs/**
   - Never place docs at root level
   - Always categorize in appropriate subdirectory

3. **No duplicate files**
   - Single source of truth for each document
   - Use symlinks or references if needed

4. **Consistent naming**
   - Follow established conventions
   - Use descriptive, clear names

Version Control

What to commit:
- Source code
- Documentation (Markdown)
- Scripts
- Configuration templates
- README, LICENSE

What NOT to commit:
- User configuration (`config.json`)
- Generated PDFs (auto-generated)
- `__pycache__/` directories
- Virtual environments
- IDE-specific files
- Backup files (`.py~`, etc.)

See `.gitignore` for complete list.

Cross-Project Consistency

This structure matches the FlexibleDietMenu project:

Similarities:
- `docs/` directory with categorized subdirectories
- Both `.md` and `.pdf` versions of documentation
- Application code in dedicated directory
- Clean root level with only essential files
- Similar documentation categories

Differences:
- This project: `protondrive_sync/` (Python package)
- FlexibleDietMenu: `nextjs_space/` (Next.js app)

Navigation Guide

For End Users

1. Start with `README.md`
2. Follow installation guide: `docs/setup/INSTALLATION_GUIDE.md`
3. Configure ProtonDrive: `docs/setup/PROTONDRIVE_SETUP.md`

4. Learn features: `docs/features/`

For Developers

1. Read `README.md` for overview
2. Review this file (`docs/STRUCTURE.md`)
3. Explore `protondrive_sync/` source code
4. Check `setup.py` for package configuration

For System Administrators

1. Review `docs/setup/INSTALLATION_GUIDE.md`
2. Check `install.sh` script
3. Read `docs/deployment/` for platform-specific info
4. Review `verify_setup.sh` for validation

For Package Maintainers

1. Read `docs/deployment/packaging.md`
2. Check `setup.py` for dependencies
3. Review distribution-specific guides
4. Test with `verify_setup.sh`

Future Structure Considerations

Potential Additions

As the project grows, consider:

```
protondrive-sync/
├── tests/                    # Unit and integration tests
│   ├── test_config.py
│   ├── test_sync.py
│   └── ...
├── assets/                   # Icons, images
│   ├── icons/
│   └── screenshots/
├── examples/                 # Example configurations
│   └── example_configs/
└── contrib/                  # Community contributions
    └── scripts/
```

Backwards Compatibility

If structure changes:
- Update all documentation
- Provide migration guide
- Update import paths
- Test thoroughly
- Announce in release notes

See Also

- Main README (../README.md) - Project overview
- Installation Guide (setup/INSTALLATION_GUIDE.md) - Setup instructions

- Packaging Guide (deployment/packaging.md) - Distribution packaging

Questions?

If you have questions about the project structure:
1. Check this document first
2. Review relevant documentation in `docs/`
3. Open an issue on GitHub
4. Consult the community

Last Updated: December 16, 2025
Maintainer: ProtonDrive Sync Contributors