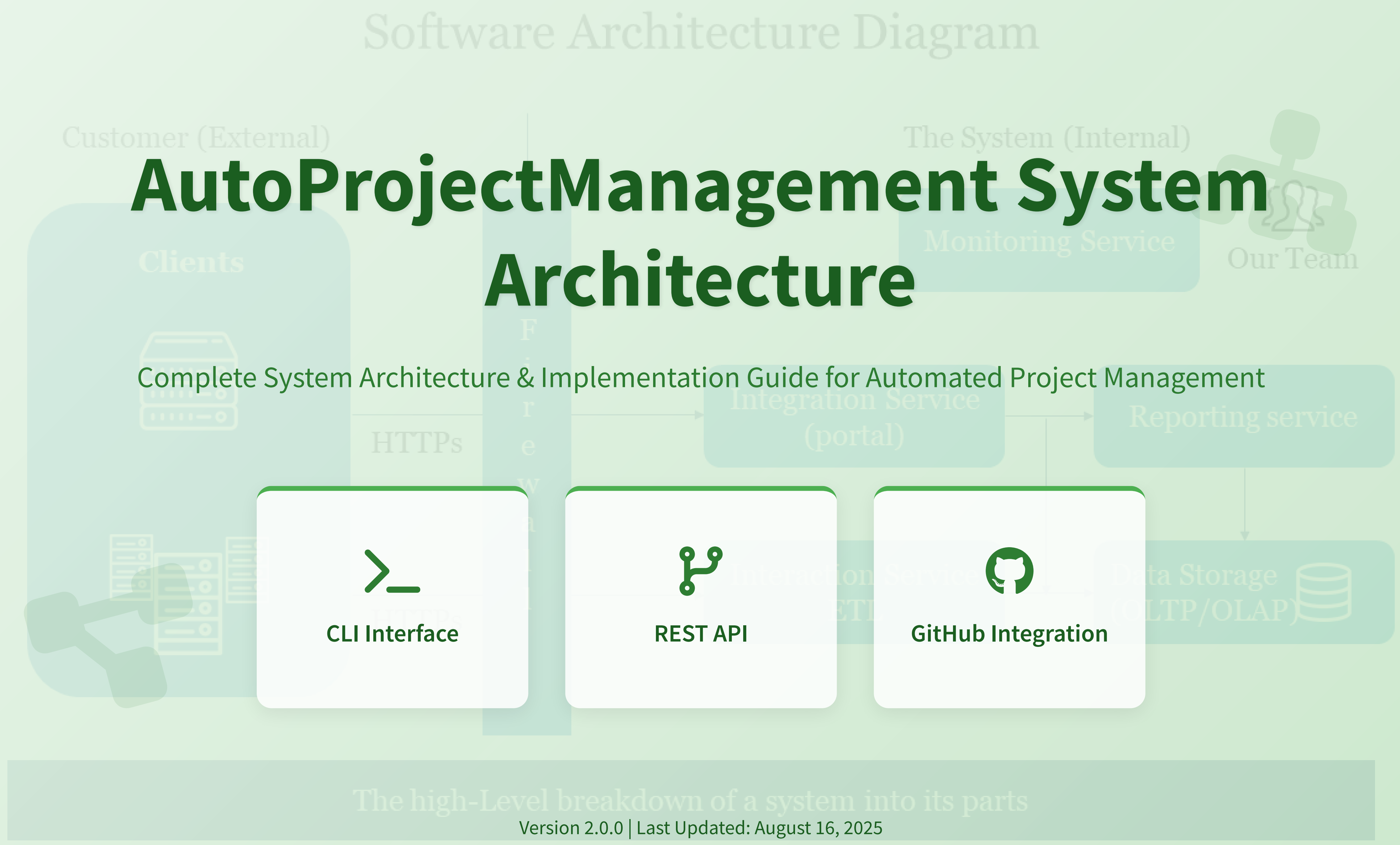

Software Architecture Diagram
Customer (External)
The System (Internal)
Clients
Monitoring Service
Our Team
Firewall
HTTPs
Integration Service (portal)
Reporting service
Information Service (ETL)
Data Storage (OLTP/OLAP)
HTTPs
The high-Level breakdown of a system into its parts

AutoProjectManagement System Architecture

Complete System Architecture & Implementation Guide for Automated Project Management

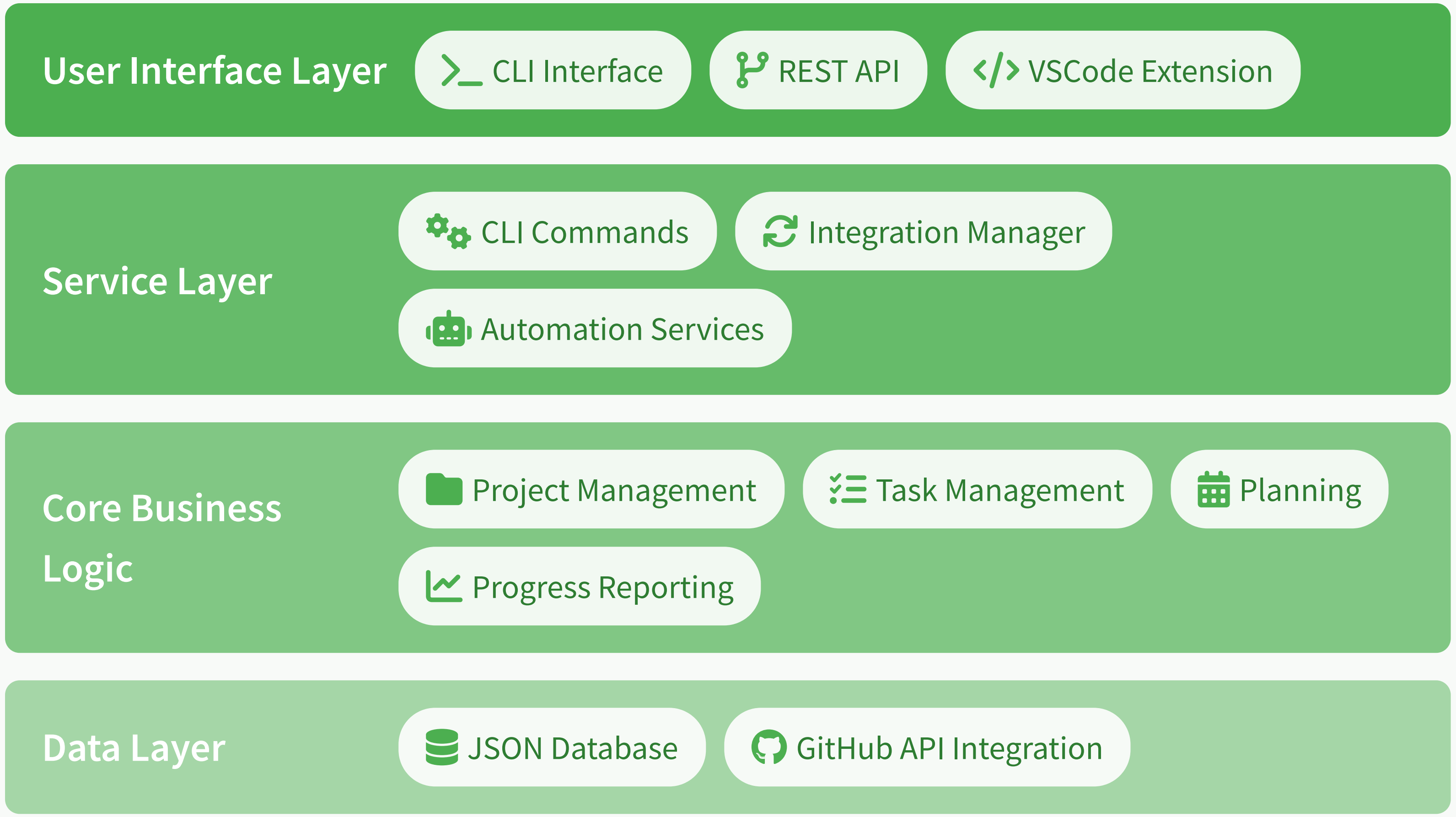
CLI Interface

REST API

GitHub Integration

Version 2.0.0 | Last Updated: August 16, 2025

System Overview & Architecture



AutoProjectManagement System

A comprehensive Python-based system with CLI and API for automated project management, task tracking, and GitHub integration.

- ✓ Powerful Command-Line Interface
- ✓ REST API for Integration
- ✓ JSON Data Persistence
- ✓ GitHub Synchronization

Technology Stack

AutoProjectManagement is built with a collection of **modern** and **optimized** technologies

Python

Version:	3.8+
Type:	Core Language

Purpose:

Core implementation

Click

Version:	8.0+
Type:	CLI Framework

Purpose:

Command-line interface

FastAPI

Version:	0.68+
Type:	API Framework

Purpose:

REST API endpoints

JSON Files

Version:	Native
Type:	Data Storage

Purpose:

Data persistence

httpx

Version:	0.24+
Type:	HTTP Client

Purpose:

GitHub API calls

pytest

Version:	7.0+
Type:	Testing Framework

Purpose:

Unit & integration tests

CLI Interface Architecture

AutoProjectManagement features a **powerful command-line interface** using the Click framework for complete project management

init

Initialize system configuration

Parameter: `--config-path`

Example:
```
apm init --config-path ./config
```

create-project

Create new project

Parameter: `project_name`

Example:
```
apm create-project "MyProject"
```

add-task

Add task to project

Parameter: `--project, --title`

Example:
```
apm add-task --project MyProject --title "Implement feature"
```

status

Show project status

Parameter: `--project`

Example:
```
apm status --project MyProject
```

report

Generate reports

Parameter: `--type, --format`

Example:
```
apm report --type progress --format markdown
```

sync-github

Sync with GitHub

Parameter: `--project`

Example:
```
apm sync-github --project MyProject
```

Core Project Management System

Core Capabilities

Project Management

Create, read, update, delete projects

`add_project()` `remove_project()` `update_project()` `get_project()`

Task Management

Manage tasks within projects

`add_task_to_project()` `remove_task_from_project()` `update_task_in_project()`

Data Validation

Ensure data integrity

`Built-in validation` `Field checking`

JSON Persistence

Store data in JSON format

`Auto serialization` `Auto deserialization`

Data Structure

Project Structure

```
{ "id": int, "name": str, "description": str,
"created_date": str, "updated_date": str,
"status": str, "priority": str, "owner": str,
"tags": List[str] }
```

Task Structure

```
{ "id": int, "title": str, "description": str,
"project_id": int, "status": str, "priority":
str, "assignee": str, "due_date": str,
"created_date": str, "updated_date": str,
"tags": List[str] }
```

API Architecture

AutoProjectManagement features a **powerful REST API** using FastAPI framework for seamless system integration

GET `/api/projects`

List all projects

Request:	`-`
Response:	`JSON array`

POST `/api/projects`

Create new project

Request:	`JSON project object`
Response:	`JSON project`

GET `/api/projects/{id}`

Get project by ID

Request:	`-`
Response:	`JSON project`

PUT `/api/projects/{id}`

Update project

Request:	`JSON project object`
Response:	`JSON project`

DELETE `/api/projects/{id}`

Delete project

Request:	`-`
Response:	`JSON confirmation`

GET `/api/projects/{id}/tasks`

List project tasks

Request:	`-`
Response:	`JSON array`

POST `/api/projects/{id}/tasks`

Add task to project

Request:	`JSON task object`
Response:	`JSON task`

Directory Structure

```
JSonDataBase/
  Inputs/
    UserInputs/
      project_config.json
      user_preferences.json
    SystemGeneratorInputs/
      system_defaults.json
  OutPuts/
    commit_progress.json
    commit_task_database.json
    progress_report.md
    project_data.json
  Backups/
    backup_*.json
    metadata/
      backup_*.json
```

Data Files

commit_progress.json

Track commit progress

```
{
  project_id: {
    task_id: status
  }
}
```

commit_task_database.json

Task database

```
{
  tasks: [task_objects]
}
```

project_data.json

Main project data

```
{
  projects: [project_objects]
}
```

progress_report.md

Markdown progress reports

```
Human-readable reports
```

GitHub Integration

AutoProjectManagement features **complete GitHub integration** that enables synchronization of projects, tasks, and documentation

Repository Creation

Create GitHub repositories for projects

`POST /user/repos`

Project Boards

Create project boards for task management

`POST /repos/{owner}/{repo}/projects`

Issue Managemen

Create and update GitHub issu

`POST /repos/{owner}/{rep`

Milestone Management

Create project milestones

`POST /repos/{owner}/{repo}/milestones`

Label Management

Create and manage issue labels

`POST /repos/{owner}/{repo}/labels`

Auto Sync

Automatic data synchronizatio

`Scheduled & event-b`

Automation Services

AutoProjectManagement features **powerful automation services** that enhance efficiency and productivity

Auto-commit

Automatic git commits

Trigger	Output
Scheduled or on change	Git commits with progress

Backup Manager

Automated backups

Trigger	Output
Scheduled	ZIP backups with metadata

Wiki Sync

Documentation synchronization

Trigger	Output
On project update	Updated wiki pages

Installation & Configuration

Installation Methods

1 pip install

```
pip install autoprojectmanagement
```

2 From source

```
git clone
https://github.com/autoprojectmanagement/autoprojectmanagement.git
cd autoprojectmanagement
pip install -e .
```

3 Development setup

```
git clone
https://github.com/autoprojectmanagement/autoprojectmanagement.git
cd autoprojectmanagement
pip install -r requirements-dev.txt
python setup.py develop
```

Configuration Setup

Initialize system

Set up the project management system

```
apm init
```

Configure GitHub

Set up API token for GitHub integration

```
apm config set
github.token
YOUR_GITHUB_TOKEN
```

Set data directory

Specify where to store project data

```
apm config set
data.path
./my_project_data
```

Future Roadmap

Planned Features

Database Support
High Priority · Q1 2025

SQLite/PostgreSQL backend

Web UI
Medium Priority · Q2 2025

React-based web interface

Mobile App
Low Priority · Q3 2025

React Native mobile app

AI Integration
Medium Priority · Q4 2025

AI-powered task suggestions

Technical Debt

Error Handling
High Impact · High Priority

Comprehensive exception handling

Logging
Low Impact · Medium Priority

Structured logging implementation

Testing Coverage
High Impact · High Priority

Achieve 90%+ test coverage

Documentation
Medium Impact · Medium Priority

Complete API documentation

Summary & Conclusion

AutoProjectManagement is a comprehensive, production-ready solution for automated project management with the following key characteristics:

Modular Architecture

Clean separation of concerns with well-defined interfaces

Multiple Interfaces

CLI, REST API, and VSCode extension support

GitHub Integration

Full synchronization with GitHub repositories and projects

JSON Persistence

Simple yet effective data storage solution

Extensible Design

Easy to add new features and integrations

Production Ready

Comprehensive testing, error handling, and documentation

This architecture document provides a complete blueprint for understanding, implementing, and extending the AutoProjectManagement system based on the actual codebase and implementation.