

سیستم مدیریت پروژه خودکار

مدیریت هوشمند پروژه‌ها با نظارت خودکار و یکپارچه‌سازی با گیت‌هاب

مستندات کامل 📄

پوشش تست 85%+ ✅

لایسنس MIT ⚡

پایتون 3.8+ 🐍



+15K

خط کد



24

ماژول اصلی



+50

گزینه پیکربندی



+25

نقطه پایانی API



نظارت زنده



همگام‌سازی خودکار

نمای کلی سیستم

سیستم جامع مدیریت پروژه خودکار مبتنی بر پایتون

آمار سیستم 📊

کل ماژول‌ها	9 اصلی + 15 زیرماژول
خطوط کد	~15,000+
پوشش تست	+85%
نقاط پایانی API	+25
گزینه‌های پیکربندی	+50

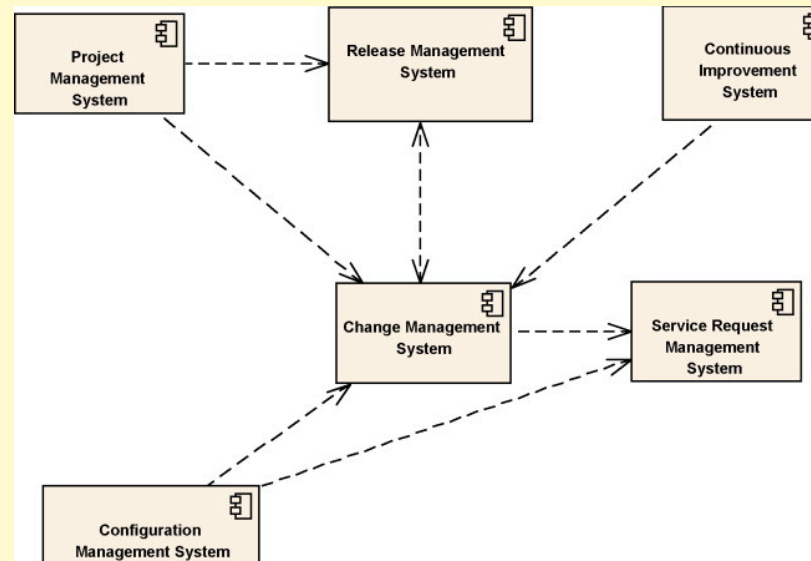
ویژگی‌های کلیدی ⭐

مدیریت پروژه 100% خودکار 🛠️	ارزیابی ریسک به صورت زنده 📊
تخصیص هوشمند منابع 🧠	پیگیری مستمر پیشرفت 💻
یکپارچه‌سازی با گیت‌هاب 📁	مستندسازی خودکار ویکی 📄
پیکربندی مبتنی بر JSON {}	رابط کاربری مبتنی بر CLI 🖥️

معماری و طراحی

ساختار سیستم و جریان داده‌ها

معماری سطح بالای سیستم



سرویس‌های یکپارچه

یکپارچه‌سازی گیت‌هاب

پایگاه داده JSON

الگوریتم‌های یادگیری ماشین

تولیدکننده گزارش

ماژول‌های اصلی

PMS - سیستم مدیریت پروژه

ماژول ریسک ارتباطی

جمع‌آوری و پردازش داده

برنامه‌ریزی و تخمین

جریان داده



ذخیره‌سازی



به‌روزرسانی پیشرفت



محاسبه ریسک



تحلیل تأثیر



تشخیص تغییرات



ورودی فایل‌ها

ماژول‌های اصلی

اجزای اصلی سیستم مدیریت پروژه خودکار

ماژول ریسک ارتباطی



شناسایی و کاهش ریسک‌های ارتباطی در پروژه

⚠️ **تضادهای کد** - تشخیص از طریق فرکانس ادغام

⚙️ **انزوای دانش** - تشخیص از طریق توزیع کامیت

🕒 **شکاف‌های ارتباطی** - زمان پاسخ به مسائل

سیستم مدیریت پروژه



رهبر اصلی که تمام عملیات پروژه را مدیریت می‌کند

✔️ **ProjectManagementSystem** - کنترلر اصلی

✔️ **Project** - ساختار داده پروژه

✔️ **Task** - مدیریت وظایف

گزارش‌دهی پیشرفت



تولید گزارش‌های جامع پیشرفت پروژه

📅 **روزانه** - هر 24 ساعت (Markdown)

📅 **هفتگی** - هر 7 روز (PDF/HTML)

📅 **ماهانه** - هر 30 روز (PDF/HTML)

جمع‌آوری و پردازش داده



جمع‌آوری و پردازش داده‌های پروژه از منابع مختلف

📁 تغییرات فایل سیستم

🕒 تاریخچه کامیت‌های گیت

🐛 مسائل و درخواست‌های گیت‌هاب

سایر ماژول‌های کلیدی

👥 مدیریت منابع

⚙️ مدیریت کیفیت کامیت

🕒 برنامه‌ریزی و تخمین

💎 درگاه API

🔧 ماژول‌های ابزاری

📊 مدیریت گردش کار وظایف

مدیریت داده

ساختار ذخیره سازی و سازماندهی فایل ها

معماری ذخیره سازی

1 فایل های JSON - ذخیره سازی اصلی داده ها

2 کش حافظه - دسترسی سریع به داده ها

3 موتور پردازش - تحلیل و پردازش داده ها

4 گیت - کنترل نسخه و ردیابی تاریخچه

لایه های ذخیره سازی سیستم

ساختار فایل ها

پیکربندی: تنظیمات سیستم و ماژول ها در فایل های JSON

داده ها: اطلاعات پروژه ها، وظایف و تحلیل ها

گزارش ها: گزارش های روزانه، هفتگی و ماهانه

پشتیبان: نسخه های پشتیبان روزانه و هفتگی

ساختار پوشه ها

```
.auto_project/
├── config/
│   ├── auto_config.json
│   └── module_configs/
├── data/
│   ├── projects.json
│   ├── tasks.json
│   └── analytics.json
├── logs/
├── reports/
└── backups/
```

اسکیمای پیکربندی JSON

id رشته

name رشته

status انتخابی

priority انتخابی

team_members آرایه

milestones آرایه آبجکت

```
{ "project": { "id": "unique_identifier", "name": "Project Name", "description": "Project description", "start_date": "2024-01-01", "end_date": "2024-12-31", "status": "active|paused|completed", "priority": "high|medium|low", "team_members": ["member1", "member2"], "milestones": [ { "id": "milestone_1", "name": "Phase 1 Complete", "target_date": "2024-06-01", "status": "pending" } ] } }
```

نصب و راه اندازی

مراحل نصب و پیکربندی اولیه سیستم

پیش نیازها

< > پایتون 3.8 یا بالاتر

🔄 گیت نصب و پیکربندی شده

👤 حساب گیت هاب با دسترسی به مخزن

روش های نصب

نصب از PyPI

(توصیه شده)

```
pip install  
autoprojectmanagement
```

نصب از سورس

```
# کلون مخزن  
git clone https://github.com/your-  
username/AutoProjectManagement.git  
cd AutoProjectManagement
```

```
# ایجاد محیط مجازی  
python -m venv venv  
source venv/bin/activate #  
venv\Scripts\activate :ویندوز
```

```
# نصب وابستگی ها  
pip install -r requirements.txt
```

```
# نصب پکیج  
pip install -e .
```

نصب با داکر

```
# ساخت ایمیج داکر  
docker build -t  
autoprojectmanagement .  
  
# اجرا در کانتینر  
docker run -v  
$(pwd)/my_project:/app/project  
autoprojectmanagement
```

پیکربندی اولیه

1 autoproject init

مقداردهی اولیه پروژه جدید

2 autoproject config --
github-token YOUR_TOKEN

پیکربندی یکپارچه سازی با گیت هاب

3 autoproject setup --
project-name "MyProject"

راه اندازی پروژه

راهنمای استفاده

دستورات پایه و گزینه‌های پیکربندی

همگام‌سازی ویکی



**autoproject wiki sync --repo-owner user -
-repo-name repo**



همگام‌سازی مستندات با ویکی گیت‌هاب

autoproject wiki sync --dry-run



پیش‌نمایش تغییرات (آزمایشی)

autoproject wiki sync --force



اجبار به همگام‌سازی کامل

دستورات مدیریت پروژه



autoproject init --name "MyProject"



ایجاد پروژه جدید

**autoproject task add --title "Feature X"
--priority high**



افزودن وظیفه جدید

autoproject status



نمایش وضعیت پروژه

autoproject report generate --type weekly



تولید گزارش پیشرفت

پیکربندی پروژه (project.json)



```
{ "project": { "name": "MyProject", "description":  
"Project description", "start_date": "2024-01-01",  
"end_date": "2024-12-31", "team_members":  
["developer1", "developer2"] }, "tasks": [],  
"milestones": [] }
```

پیکربندی اصلی (auto_config.json)



```
{ "system": { "check_interval": 300,  
"auto_commit_threshold": 5, "report_interval":  
86400 }, "github": { "token": "your_github_token",  
"repo_owner": "your-username", "repo_name": "your-  
repo" }, "notifications": { "slack_webhook":  
"https://hooks.slack.com/...", "email_enabled":
```

<> استفاده پیشرفته

یکپارچه‌سازی API



استفاده از **GitHubIntegration** برای تعامل با API گیت‌هاب و ایجاد مسائل و درخواست‌ها

توسعه ماژول سفارشی



ایجاد ماژول‌های سفارشی با ارث‌بری از **BaseModule** و پیاده‌سازی متد **(process)**

انواع تست‌ها



UnitTests



تست‌های واحد برای ماژول‌های مجزا

IntegrationTests



تست‌های یکپارچگی بین ماژول‌ها

SystemTests



تست‌های سیستم کامل

PerformanceTests



تست‌های عملکرد و سرعت

SecurityTests



تست‌های امنیتی

اجرای تست‌ها



```
pip install -r requirements-dev.txt
```

نصب وابستگی‌های تست

```
pytest tests/
```



اجرای تمام تست‌ها

```
pytest tests/code_tests/UnitTests/
```



اجرای دسته تست خاص

```
pytest --cov=autoprojectmanagement tests/
```



اجرای تست با پوشش

```
pytest tests/code_tests/IntegrationTests/
```



اجرای تست‌های یکپارچگی

ساختار تست‌ها



UnitTests: تست اجزای مجزا سیستم



IntegrationTests: تست تعامل بین ماژول‌ها



conftest.py: تنظیمات و fixtureهای مشترک



pytest.ini: پیکربندی pytest



```
tests/
├── code_tests/
│   ├── 01_UnitTests/
│   ├── 02_IntegrationTests/
│   ├── 03_SystemTests/
│   ├── 04_PerformanceTests/
│   └── 05_SecurityTests/
├── conftest.py
├── pytest.ini
└── README.md
```

<> نوشتن تست‌ها

```
import pytest
from autoprojectmanagement.main_modules import ProjectManagementSystem

def test_project_creation():
    pms = ProjectManagementSystem()
    project = pms.create_project("Test Project")
    assert project.name == "Test Project"
    assert project.status == "active"
```