سیستم مدیریت پروژه خودکار

مدیریت هوشمند پروژه‌ها با نظارت خودکار و یکپارچه‌سازی با گیت‌هاب

پایتون 3.8+ | لایسنس MIT | پوشش تست 85%+ | مستندات کامل

24
ماژول اصلی

+15K
خط کد

+25
نقطه پایانی API

+50
گزینه پیکربندی

همگام‌سازی خودکار

نظارت زنده

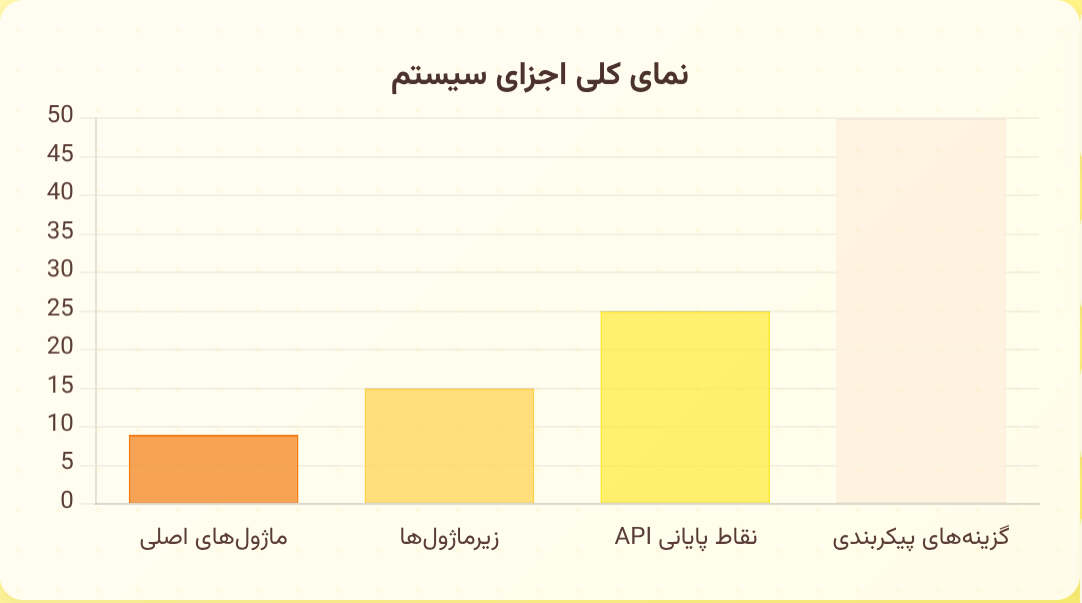
نمای کلی سیستم

سیستم جامع مدیریت پروژه خودکار مبتنی بر پایتون

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

آمار سیستم

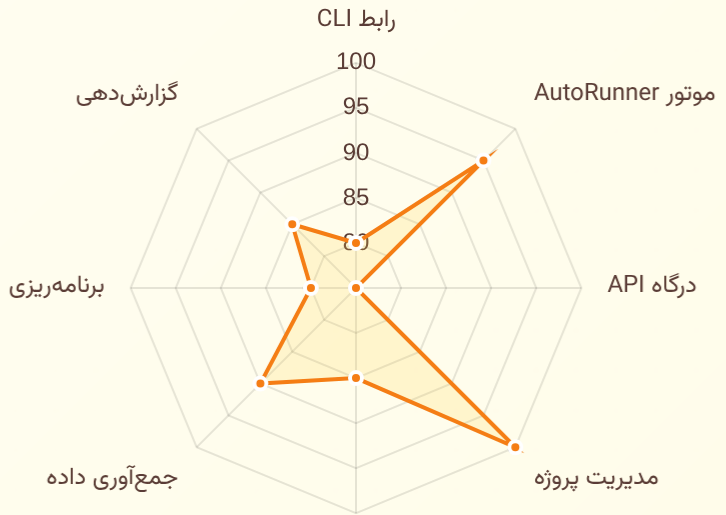


24	+15K	+85%
ماژول اصلی	خط کد	پوشش تست

معماری و طراحی

ساختار سیستم و جریان داده‌ها

معماری سطح بالای سیستم



ماژول‌های اصلی

- PMS - سیستم مدیریت پروژه
- ماژول ریسک ارتباطی
- جمع‌آوری و پردازش داده
- برنامه‌ریزی و تخمین

سرویس‌های یکپارچه

- یکپارچه‌سازی گیت‌هاب
- پایگاه داده JSON
- الگوریتم‌های یادگیری ماشین
- تولیدکننده گزارش

جریان داده

ورودی فایل‌ها — تشخیص تغییرات — تحلیل تأثیر — محاسبه ریسک — به‌روزرسانی پیشرفت — ذخیره‌سازی

ماژول‌های اصلی

اجزای اصلی سیستم مدیریت پروژه خودکار

سیستم مدیریت پروژه

رهبر اصلی که تمام عملیات پروژه را مدیریت می‌کند

- **ProjectManagementSystem** - کنترلر اصلی
- **Project** - ساختار داده پروژه
- **Task** - مدیریت وظایف

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

جمع‌آوری و پردازش داده

جمع‌آوری و پردازش داده‌های پروژه از منابع مختلف

- تغییرات فایل سیستم
- تاریخچه کامیت‌های گیت
- مسائل و PRهای گیت‌هاب

گزارش‌دهی پیشرفت

تولید گزارش‌های جامع پیشرفت پروژه

- **روزانه** - هر 24 ساعت (Markdown)
- **هفتگی** - هر 7 روز (PDF/HTML)
- **ماهانه** - هر 30 روز (PDF/HTML)

سایر ماژول‌های کلیدی

- برنامه‌ریزی و تخمین
- مدیریت کیفیت کامیت
- مدیریت منابع
- مدیریت گردش کار وظایف
- ماژول‌های ابزاری
- درگاه API

مدیریت داده

ساختار ذخیره‌سازی و سازماندهی فایل‌ها

معماری ذخیره‌سازی

1. **فایل‌های JSON** - ذخیره‌سازی اصلی داده‌ها
2. **کش حافظه** - دسترسی سریع به داده‌ها
3. **موتور پردازش** - تحلیل و پردازش داده‌ها
4. **گیت** - کنترل نسخه و ردیابی تاریخچه

ساختار فایل‌ها

ساختار پوشه‌ها

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

- **پیکربندی:** تنظیمات سیستم و ماژول‌ها در فایل‌های JSON
- **داده‌ها:** اطلاعات پروژه‌ها، وظایف و تحلیل‌ها
- **گزارش‌ها:** گزارش‌های روزانه، هفتگی و ماهانه
- **پشتیبان:** نسخه‌های پشتیبان روزانه و هفتگی

</> اسکیمای پیکربندی JSON

```
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" } ] }
```

- **id** رشته
- **name** رشته
- **status** انتخابی
- **priority** انتخابی
- **team_members** آرایه
- **milestones** آرایه آبجکت

نصب و راه‌اندازی

مراحل نصب و پیکربندی اولیه سیستم

پیش‌نیازها

- پایتون 3.8 یا بالاتر
- گیت نصب و پیکربندی شده
- حساب گیت‌هاب با دسترسی به مخزن

روش‌های 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 # ویندوز:
venv\Scripts\activate

# نصب وابستگی‌ها
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 یکپارچه‌سازی با گیت‌هاب
3. `autoproject setup --project-"name "MyProject`
   راه‌اندازی پروژه

راهنمای استفاده

دستورات پایه و گزینه‌های پیکربندی

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

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

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": true }
```

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
```

استفاده پیشرفته

توسعه ماژول سفارشی

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

یکپارچه‌سازی API

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

تست

نحوه اجرای تست‌ها و ساختار تست سیستم

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

انواع تست‌ها

UnitTests
تست‌های واحد برای ماژول‌های مجزا

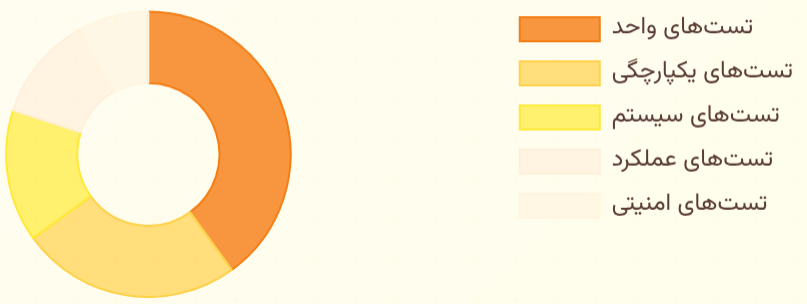
IntegrationTests
تست‌های یکپارچگی بین ماژول‌ها

SystemTests
تست‌های سیستم کامل

PerformanceTests
تست‌های عملکرد و سرعت

SecurityTests
تست‌های امنیتی

توزیع تست‌ها




```
/tests
/code_tests ──┤
/UnitTests_01 ──┤   │
/IntegrationTests_02 ──┤   │
/SystemTests_03 ──┤   │
/PerformanceTests_04 ──┤   │
/SecurityTests_05 ──└   │
conftest.py ──┤
pytest.ini ──┤
README.md ──└
```

UnitTests: تست اجزای مجزا سیستم

IntegrationTests: تست تعامل بین ماژول‌ها

conftest.py: تنظیمات و fixtureهای مشترک

pytest.ini: پیکربندی pytest

</> نوشتن تست‌ها

```python
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"
```