مستندات فنی پروژه

۱ .مقدمه و هدف پروژه

هدف از این پروژه طراحی و پیاده‌سازی یک سامانه مدیریت منابع پردازشی بر پایه مدل **GPU-as-a-Service (GaaS)** است. این سامانه به مدیران اجازه می‌دهد سهمیه زمانی مشخصی (بر حسب ساعت) به کاربران اختصاص دهند. کاربران نیز می‌توانند درخواست‌های پردازشی خود را ثبت کنند؛ سیستم به طور خودکار سهمیه آن‌ها را چک کرده و در صورت مجاز بودن، زمان درخواستی را کسر و تسک را ثبت می‌کند.

۲ .تکنولوژی‌های مورد استفاده

در توسعه این سامانه از جدیدترین ابزارهای دنیای برنامه‌نویسی پایتون استفاده شده است:

- **FastAPI:** فریم‌ورک مدرن و سریع برای ساختAPI.
- **SQLAlchemy:** کتابخانه ORM برای مدیریت ارتباط با پایگاه داده به صورت شیءگرا.
- **SQLite:** پایگاه داده سبک و منعطف برای ذخیره‌سازی داده‌ها.
- **Pydantic:** جهت اعتبارسنجی داده‌های ورودی.
- **Docker:** کانتینرایز کردن پروژه جهت اجرای یکپارچه در تمامی محیط‌ها.

۳ .معماری دیتابیس و مدل‌ها

دیتابیس پروژه شامل دو جدول اصلی با رابطه یک‌به‌چند (One-to-Many) است:

۳.۱ .مدل کاربر(User)

این مدل شامل اطلاعات هویتی و میزان سهمیه کاربر است:

- Username: نام کاربری یکتا.
- password_hash: رمز عبور که به صورت هش‌شده ذخیره می‌شود.

- Role: نقش کاربر (ادمین یا کاربر عادی).
- gpu_hours_quota: سهمیه باقی‌مانده کاربر که به صورت پیش‌فرض ۱۰ ساعت در نظر گرفته شده است.

۳.۲ .مدل تسک(Job)

هر تسک شامل جزئیات فنی اجرای عملیات است:

- command: دستور اجرایی پایتون یا.Bash
- gpu_type: نوع کارت گرافیک درخواستی (مثلاً NVIDIA A100).
- Status: وضعیت تسک (PENDING, APPROVED, RUNNING, ...) که از نوع **Enum** تعریف شده است.
- user_id: کلید خارجی متصل به جدول کاربران.

عکس های فایل models.py

```python
from sqlalchemy import Column, Integer, String, Float, ForeignKey, Enum
from sqlalchemy.orm import relationship, declarative_base
import enum

Base = declarative_base()

class JobStatus(enum.Enum):
    PENDING = "PENDING"
    APPROVED = "APPROVED"
    RUNNING = "RUNNING"
    COMPLETED = "COMPLETED"
    FAILED = "FAILED"

class User(Base):
    __tablename__ = "users"
    id = Column(Integer, primary_key=True)
    username = Column(String, unique=True)
    password_hash = Column(String)
    role = Column(String) # نقش کاربر: admin یا user
    gpu_hours_quota = Column(Float, default=10.0) # سهمیه ساعت کاربر

class Job(Base):
    __tablename__ = "jobs"
    id = Column(Integer, primary_key=True)
    command = Column(String) # دستور اجرا
    gpu_type = Column(String) # نوع GPU
    estimated_hours = Column(Float) # تخمین زمان اجرا
    status = Column(Enum(JobStatus), default=JobStatus.PENDING) # وضعیت تسک
    user_id = Column(Integer, ForeignKey("users.id"))

    owner = relationship("User")
```

۴. امنیت و احراز هویت

امنیت سامانه بر پایه پروتکل **JWT (JSON Web Token)** پیاده‌سازی شده است.

- **هشینگ:** برای سازگاری کامل با پایتون ۳.۱۴ و امنیت بالا، از الگوریتم **PBKDF2-SHA256** جهت ذخیره‌سازی رمزهای عبور استفاده شده است.
- **توکن:** پس از ورود موفقیت‌آمیز، یک توکن با اعتبار ۳۰ دقیقه برای کاربر صادر می‌شود که شامل اطلاعات نقش (Role) اوست.

عکس های فایل auth.py

```python
from passlib.context import CryptContext
from datetime import datetime, timedelta, timezone
from jose import JWTError, jwt

# تنظیمات امنیتی
SECRET_KEY = "a_very_secret_key_12345"
ALGORITHM = "HS256"
ACCESS_TOKEN_EXPIRE_MINUTES = 30

# 3.14 تغییر متد هش برای سازگاری با پایتون
pwd_context = CryptContext(schemes=["pbkdf2_sha256"], deprecated="auto")

def get_password_hash(password):
    return pwd_context.hash(password)

def verify_password(plain_password, hashed_password):
    return pwd_context.verify(plain_password, hashed_password)

def create_access_token(data: dict):
    to_encode = data.copy()
    expire = datetime.now(timezone.utc) + timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES)
    to_encode.update({"exp": expire})
    encoded_jwt = jwt.encode(to_encode, SECRET_KEY, algorithm=ALGORITHM)
    return encoded_jwt
```

۵ .منطق عملکرد برنامه(Business Logic)

اصلی‌ترین بخش پروژه، مدیریت هوشمند سهمیه در زمان ثبت تسک است :۱ .سیستم ابتدا کاربر را شناسایی می‌کند .۲ .میزان ساعت درخواستی با سهمیه موجود مقایسه می‌شود .۳ .اگر سهمیه کافی نباشد، خطای **400 Bad Request**صادر می‌شود .۴ .در صورت تایید، میزان ساعت بلافاصله از سهمیه کاربر کسر شده و تسک با وضعیت "در انتظار تایید (PENDING) "ثبت می‌گردد.

۶ .معرفی مسیرهای(API (Endpoints

سامانه دارای مسیرهای زیر برای تعامل با کاربر است:

- POST /register: ثبت‌نام کاربر جدید.
- POST /login: ورود و دریافت توکن امنیتی.
- POST /jobs: ثبت تسک جدید و کسر سهمیه (مخصوص کاربر.(
- PATCH /jobs/{id}: تغییر وضعیت تسک توسط ادمین.

۷ .راهنمای نصب و اجرا

برای اجرای پروژه، ابتدا کتابخانه‌های مورد نیاز را نصب کنید:

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

سپس سرور را با دستور زیر اجرا نمایید:

```
uvicorn main:app --reload
```

همچنین پروژه با استفاده از **Docker Compose**قابل اجراست:

```
docker-compose up --build
```