به نام خدا


# دانشکده مهندسی کامپیوتر

## مهندسی نرم افزار ۱

## "فاز 7 پروژه"

## موضوع پروژه:
## پیاده‌سازی

**استاد درس: دکتر کلباسی**

مهلت تحویل: باتوجه به مهلت ثبت موقت نمرات پایانترم، اعلام خواهد شد.

نیم‌سال اول ۱۴۰۴

## ۱. معرفی کلی پروژه و ساختار پایه معماری

این پروژه یک سامانه‌ی وب چندبخشی با موضوع نقشه و جغرافیای ایران است که در قالب یک معماری سرویس‌گرا طراحی شده است. در این سامانه، مجموعه‌ای از خدمات مستقل در کنار یکدیگر قرار می‌گیرند که هر کدام وظیفه‌ی مشخصی را بر عهده دارند و در مجموع یک ابزار جامع برای مدیریت، پردازش و بهبود داده‌ها فراهم می‌کنند.

هر تیم مسئول طراحی و پیاده‌سازی یک خدمت یا میکروسرویس مشخص است. این میکروسرویس‌ها به‌صورت مستقل توسعه داده می‌شوند اما همگی باید به سامانه‌ی مرکزی متصل باشند تا خروجی نهایی پروژه به‌صورت یکپارچه در اختیار کاربر قرار گیرد. سامانه‌ی مرکزی توسط تیم پایه‌ی پروژه پیاده‌سازی شده است و تیم‌ها صرفاً وظیفه‌ی توسعه‌ی سرویس‌های اختصاصی خود را دارند.

در پیاده‌سازی سرویس‌ها، تیم‌ها در انتخاب زبان برنامه‌نویسی و فریم‌ورک آزادی عمل دارند. در صورتی که سرویس تیم خارج از Django توسعه داده شود، لازم است ارتباط میان میکروسرویس خارجی و سامانه‌ی مرکزی Django برقرار شود. این ارتباط می‌تواند از روش‌های مختلفی انجام شود؛ از جمله استفاده از gRPC یا message broker هایی مانند RabbitMQ، انتخاب روش ارتباطی به عهده‌ی تیم است، اما باید با معماری کلی پروژه سازگار باشد.

## ۲. نقش و مسئولیت سامانه‌ی مرکزی (Django Core)

سامانه‌ی مرکزی Django به‌عنوان هسته‌ی اصلی پروژه عمل می‌کند و مسئولیت قابلیت‌های مشترک میان تمام تیم‌ها را بر عهده دارد. این سامانه وظایف زیر را انجام می‌دهد:

- مدیریت احراز هویت کاربران شامل ثبت‌نام، ورود و خروج
- نگهداری وضعیت احراز هویت کاربر از طریق Session و Cookie
- ارائه‌یAPI های مشترک برای بررسی وضعیت کاربر
- نگهداری ساختار کلی پروژه و مسیرهای اصلی
- فراهم‌سازی یک نقطه‌ی اتصال استاندارد برای سرویس‌های تیمی

تمام درخواست‌هایی که به سامانه‌ی مرکزی Django می‌رسند، باید به یک مسیر مشخص (Sub-URL) هدایت شوند که مربوط به سرویس تیم است. هر تیم می‌تواند API های اختصاصی خود را در همین مسیرها تعریف و مدیریت کند.

## ۳. ساختار کلی پروژه و محل توسعه تیم‌ها

در شاخه‌ی اصلی ریپازیتوری، برای هر تیم یک فولدر مجزا در نظر گرفته شده است. هر تیم موظف است تمام تغییرات، کدها و فایل‌های مربوط به پروژه‌ی خود را صرفاً در فولدر اختصاصی همان تیم انجام دهد.

به‌عنوان مثال، اگر تیم مربوطه Team1 باشد، تمام تغییرات باید در فولدر team1 انجام شود. تغییر ساختار کلی پروژه یا فایل‌های سایر تیم‌ها مجاز نیست و در صورت نیاز به تغییرات ساختاری، باید ابتدا هماهنگی لازم انجام شود.

برای هر تیم، یک Application جداگانه در Django در نظر گرفته شده است که با نام همان تیم نام‌گذاری شده است. همچنین فایل urls.py مربوط به آن Application در اختیار تیم قرار دارد تا مسیرهای مورد نیاز سرویس خود را در آن تعریف کند.

در صورتی که تیم قصد استفاده از Django را نداشته باشد، می‌تواند میکروسرویس خود را با هر زبان یا فریم‌ورک دیگری توسعه دهد و از طریق روش‌های ارتباطی تعریف‌شده (مانند HTTP، gRPC یا message broker) به سامانه‌ی مرکزی متصل شود.

## ۴. استفاده از Docker در معماری پروژه

Docker در این پروژه به‌عنوان بستر اصلی اجرای سرویس‌ها استفاده شده است. تمام اجزای پروژه، شامل سامانه‌ی مرکزی و سرویس‌های تیمی، در قالب کانتینرهای Docker اجرا می‌شوند. این رویکرد باعث می‌شود محیط اجرا برای تمام تیم‌ها یکسان باشد و مدیریت وابستگی‌ها و اجرای هم‌زمان سرویس‌ها ساده‌تر انجام شود.

هر تیم سرویس‌های خود را به‌صورت مستقل در Docker اجرا می‌کند و نیازی به اجرای مستقیم آن‌ها روی سیستم میزبان وجود ندارد. ارتباط میان سرویس‌ها از طریق شبکه‌ی داخلی Docker برقرار می‌شود.

## ۵. Docker Compose و نحوه‌ی اضافه کردن سرویس جدید

برای هر تیم یک فایل docker-compose.yml در فولدر همان تیم وجود دارد. این فایل محل تعریف تمام سرویس‌های تیم است. در صورت نیاز به اضافه کردن یک سرویس جدید، تیم باید مراحل زیر را انجام دهد:

- تعریف سرویس جدید در فایل docker-compose تیم
- قرار دادن سرویس در شبکه‌ی مشترک پروژه
- تعریف متغیرهای محیطی مورد نیاز سرویس
- اتصال سرویس به Gateway تیم از طریق Nginx

هیچ نیازی به تغییر docker-compose سامانه‌ی مرکزی یا تیم‌های دیگر وجود ندارد.

## ۶. نقش Nginx و منطق استفاده از آن در پروژه

در این پروژه، Nginx به‌عنوان Gateway هر تیم استفاده می‌شود .Gateway تنها نقطه‌ی ورودی تیم از بیرون است و تمام درخواست‌های مربوط به آن تیم ابتدا وارد Nginx می‌شوند.

منطق استفاده از Nginx به این صورت است که هر تیم تنها یک پورت روی سیستم میزبان دارد و این پورت به Nginx متصل است . Nginx بر اساس مسیر URL تصمیم می‌گیرد درخواست به کدام سرویس داخلی تیم هدایت شود.

در صورتی که تیم سرویس جدیدی اضافه کند، کافی است:

- سرویس را در docker-compose اضافه کند
- یک مسیر جدید برای آن سرویس در فایل تنظیمات Nginx تعریف کند

بدون آنکه پورت جدیدی روی سیستم میزبان باز شود.

## ۷. پایگاه داده و نحوه‌ی استفاده تیم‌ها

در سامانه‌ی مرکزی مبتنی بر Django ، یک پایگاه داده‌ی ابری MySQL به‌عنوان دیتابیس اصلی پروژه در نظر گرفته شده است. اطلاعات اتصال به این دیتابیس از طریق متغیرهای محیطی (Environment Variables) در اختیار هر تیم قرار می‌گیرد.

هر تیم:

- یک نام کاربری و رمز عبور اختصاصی دریافت می‌کند.
- این اطلاعات صرفاً مختص همان تیم بوده و نباید در اختیار سایر تیم‌ها قرار گیرد.
- دسترسی هر تیم محدود به جدول‌هایی است که در فازهای قبلی برای همان تیم ایجاد شده‌اند.
- هیچ تیمی امکان مشاهده یا دسترسی به جدول‌های سایر تیم‌ها را ندارد.

برای اتصال به دیتابیس مرکزی:

- کافی است متغیر مربوط به اتصال دیتابیس تیم در فایل env. تعریف شود.
- نحوه‌ی افزودن این متغیر در فایل env.example. توضیح داده شده است.
- هیچ تنظیم اضافی دیگری لازم نیست و سرویس‌های تیم می‌توانند مستقیماً از این متغیر استفاده کنند.

سطح دسترسی تیم‌ها به دیتابیس ابری شامل موارد زیر است:

- SELECT (خواندن داده‌ها)
- INSERT (افزودن داده‌ها)
- UPDATE (ویرایش داده‌ها)

در مرحله‌ی توسعه:

- تیم‌ها می‌توانند از دیتابیس محلی Django مانند SQLite استفاده کنند.
- مدل‌ها و migration ها به‌صورت محلی اجرا می‌شوند.
- اعمال migration روی دیتابیس اصلی صرفاً پس از بررسی و تأیید تیم تدریسیاری انجام می‌شود.

در صورت نیاز به:

- ایجاد جدول جدید(CREATE)
- تغییر ساختار جدول(ALTER)
- حذف جدول(DROP)

تیم‌ها باید:

۱. دستور SQL مربوطه را در بخش «پایگاه داده» این شیت ثبت کنند.
۲. توضیح مختصری درباره هدف تغییر ارائه دهند.
۳. پس از بررسی و تأیید تیم تدریسیاری، تغییر موردنظر روی دیتابیس ابری اعمال خواهد شد.

### ۷-۱. نحوه استفاده تیم‌ها از دیتابیس در کد جنگو

در تنظیمات پروژه، برای هر تیم یک اتصال پایگاه‌داده مستقل بر اساس متغیر محیطی <TEAMNAME>_DATABASE_URL تعریف شده است. برای مثال، تیم team1 از متغیر TEAM1_DATABASE_URL استفاده می‌کند.

در صورتی که این متغیر محیطی در فایل env تعریف نشده باشد، پروژه به‌صورت خودکار از یک دیتابیس محلی SQLite مخصوص همان تیم استفاده می‌کند که در مسیر فولدر تیم ایجاد می‌شود. این رفتار امکان توسعه و تست محلی را بدون نیاز به دیتابیس ابری فراهم می‌کند.

هدایت درخواست‌های دیتابیس توسط یک Router انجام می‌شود (TeamPerAppRouter) که باعث می‌شود تمام عملیات خواندن و نوشتن هر اپلیکیشن به دیتابیس مربوط به همان تیم ارسال شود و از تداخل میان تیم‌ها جلوگیری گردد.

در کد Django تیم‌ها، استفاده از ORM به‌صورت معمول کافی است و نیازی به تعیین دستی دیتابیس وجود ندارد. در صورت نیاز خاص، تیم می‌تواند به‌صورت صریح از alias مربوط به تیم خود (مثلاً team1) استفاده کند.

## ۸. اتصال سرویس‌های تیم به سامانه‌ی مرکزی

سرویس‌هایی که نیاز به ارتباط با سامانه‌ی مرکزی دارند، باید در همان شبکه‌ی Docker که Core در آن قرار دارد تعریف شوند. برای این منظور، در docker-compose تیم:

- شبکه‌ی app404_net به سرویس اضافه می‌شود
- متغیر محیطی CORE_BASE_URL مقداردهی می‌شود

این موارد به صورت پیش فرض و برای مثال در فایل docker-compose هر تیم قرار داده شده تا با نحوه استفاده از آن آشنا شوید.

به این ترتیب، سرویس تیم می‌تواند API های سامانه‌ی مرکزی را از طریق آدرس داخلی Docker فراخوانی کند، به‌صورت:

core:8000/api/...

core:8000/team1/map

این آدرس فقط در داخل شبکه‌ی Docker معتبر است و به‌عنوان یک متغیر محیطی در کد سرویس استفاده می‌شود.

## ۹. احراز هویت و هدایت درخواست‌ها

احراز هویت کاربران به‌صورت کامل در سامانه‌ی مرکزی انجام می‌شود. پس از ورود کاربر، اطلاعات احراز هویت در قالب Cookie ذخیره می‌شود. این Cookie توسط Nginx بدون تغییر به سرویس‌های تیمی منتقل می‌شود.

سرویس‌های تیمی می‌توانند با ارسال درخواست به API های احراز هویت سامانه‌ی مرکزی با استفاده از Cookie کاربر، وضعیت کاربر را بررسی کرده و در صورت نیاز اطلاعات کاربر را دریافت کنند. تیم نیازی به پیاده‌سازی منطق احراز هویت یا اعتبارسنجی توکن ندارد.

CORE_BASE_URL:8000/api/auth/me/

CORE_BASE_URL:8000/api/auth/verify/

### ۱۰. مسیرهای دسترسی و Sub-URL ها

تمام درخواست‌هایی که به سامانه‌ی مرکزی می‌رسند باید به Sub-URL مربوط به سرویس تیم هدایت شوند. هر تیم می‌تواند مسیرهای مختلفی برای سرویس خود تعریف کند.

برای مثال:

- سرویس Map:
  - /team1/map
  - /team1/map/eta
- سرویس trip plan:
  - /team2/plan
  - /team2/plan/list

با تعریف مسیرهای مختلف، تمام درخواست‌ها به میکروسرویس تیم هدایت شده و مدیریت سرویس به‌صورت کامل در اختیار تیم خواهد بود.

### ۱۱. اسکریپت‌های اجرایی پروژه و نحوه استفاده

به‌منظور ساده‌سازی اجرای پروژه و جلوگیری از خطاهای ناشی از اجرای دستی سرویس‌ها، مجموعه‌ای از اسکریپت‌های اجرایی در پروژه در نظر گرفته شده است. این اسکریپت‌ها در دو فولدر جداگانه برای سیستم‌عامل‌های Windows و Linux ارائه شده‌اند، اما منطق عملکرد آن‌ها کاملاً یکسان است.

در هر یک از این فولدرها سه اسکریپت وجود دارد که وظایف مشخصی را بر عهده دارند.

اسکریپت up-all (با پسوند .sh در Linux و .ps1 در Windows) مسئول راه‌اندازی کامل پروژه است. این اسکریپت سامانه‌ی مرکزی (Core) و تمامی سرویس‌های تیم‌ها را به‌صورت هم‌زمان اجرا کرده، پورت اختصاصی هر تیم را تنظیم می‌کند و Gateway و سرویس‌های مربوط به هر تیم را بالا می‌آورد. این اسکریپت برای زمانی مناسب است که نیاز به اجرای کل پروژه وجود دارد.

اسکریپت up-team برای اجرای یک تیم مشخص استفاده می‌شود. این اسکریپت شماره تیم را به‌عنوان ورودی دریافت کرده و تنها سرویس‌های مربوط به همان تیم را اجرا می‌کند. استفاده از این اسکریپت برای توسعه و تست روزمره‌ی هر تیم توصیه می‌شود.

اسکریپت down-all برای متوقف کردن کامل پروژه در نظر گرفته شده است. با اجرای این اسکریپت، تمامی سرویس‌های تیمی و سامانه‌ی مرکزی متوقف شده و منابع Docker آزاد می‌شوند.

استفاده از این اسکریپت‌ها بخشی از طراحی اجرایی پروژه محسوب می‌شود و اجرای پروژه بدون آن‌ها توصیه نمی‌شود؛ زیرا تنظیم پورت‌ها، ترتیب اجرای سرویس‌ها و هماهنگی میان تیم‌ها به‌صورت متمرکز در این اسکریپت‌ها مدیریت شده است.

## راه‌اندازی سریع پروژه

برای اجرای سریع پروژه کافی است از ریشه‌ی پروژه یکی از اسکریپت‌های زیر اجرا شود:

اجرای کامل پروژه:

up-all

اجرای یک تیم مشخص:

up-team

توقف کامل پروژه:

down-all

در نهایت میتوانید در مرورگر خود به آدرس http://localhost:8000/ برای مشاهده صفحه ابتدایی پروژه مراجعه کنید.

### ۱۲. استاندارد ظاهری و هماهنگی UI/UX

برای حفظ یکپارچگی ظاهری سامانه، تیم‌ها باید در طراحی رابط کاربری خود از پالت رنگی، فونت و ساختار کلی پروژه پیروی کنند. این استاندارد شامل موارد زیر است:

- استفاده از فونت اصلی پروژه(Vazirmatn)
- رعایت جهت راست‌به‌چپ
- استفاده از رنگ‌های الهام‌گرفته از طبیعت و هویت بصری ایران
- حفظ ساختار کلی Header، Footer و فاصله‌گذاری‌ها مشابه سامانه‌ی اصلی

هدف از این استاندارد، ایجاد تجربه‌ی کاربری یکپارچه است؛ به‌طوری که کاربر هنگام ورود به بخش‌های مختلف سامانه، تفاوت ناگهانی در ظاهر یا رفتار رابط کاربری احساس نکند.

برای آشنایی دقیق تر با اطلاعات مربوط به ظاهر صفحه ها میتوانید فایل style.css های موجود در پروژه را مطالعه نمایید.

## ۱۳. جمع‌بندی نهایی

در این معماری، هر تیم به‌صورت مستقل سرویس خود را توسعه می‌دهد، اما از طریق استانداردهای مشخص‌شده به سامانه‌ی مرکزی متصل می‌شود. این ساختار باعث می‌شود توسعه‌ی پروژه به‌صورت هم‌زمان، منظم و قابل مدیریت انجام شود و در نهایت خروجی تمام تیم‌ها در قالب یک سامانه‌ی واحد در اختیار کاربر قرار گیرد.

### رعایت کدنویسی تمیز

- در تمامی مراحل از نام‌گذاری‌های واضح و با معنا استفاده کنید.
- توابع در حد امکان باید کوچک باشند و تنها یک وظیفه را انجام دهند.
- از نام‌گذاری‌های طولانی نترسید؛ یک نام مناسب که مفهوم خود را توضیح دهد بهتر از نام‌های کوتاه و مبهم است.
- کدهای خود را به صورت تدریجی و به مرور زمان کامیت کنید. پیام‌های کامیت نیز باید با معنا و واضح باشند.
- سعی کنید در تیم خود قراردادهایی برای کدنویسی تعریف کنید تا خروجی نهایی یکدست و منسجم باشد.
- برای کدهای خود مستندسازی کنید تا در صورت نیاز به مراجعه در آینده، فهم آن آسان‌تر باشد.
- مشابه فایل requirements.txt که در پروژه استفاده می‌شود و با دستور زیر نیازمندی‌ها را نصب می‌کند:

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

❋ برای مایکروسرویس خود نیز یک فایل requirements تعریف کرده و تمام نیازمندی‌های لازم برای اجرای آن را در آن قرار دهید.

- برای Application خود در فولدر مربوط به گروه خود یک فایل README ایجاد کرده و توضیحات لازم را در آن قرار دهید.

## ارسال کد در گیت‌هاب

آدرس ریپازیتوری:

**https://github.com/MajidNami/Software-Engineering-1404-01_G2**

برای توضیحات کامل، به فایل README ریپازیتوری مراجعه کنید.

برای ارسال کد خود، باید مراحل زیر را انجام دهید:

۱. ریپازیتوری را Fork کنید.

۲. ریپازیتوری Fork شده را Clone کنید.

۳. تغییرات خود را در یک Branch جدید انجام دهید و سپس آن را Commit & Push کنید.

۴. درخواست Pull Request ارسال کنید.

- اگر در مورد استفاده از سرور جنگو و گیت‌هاب سوال، پیشنهاد یا مشکلی دارید، می‌توانید از طریق تلگرام یا ایمیل با راه‌های ارتباطی زیر زیر در ارتباط باشید:

@moein_enayati | moenayatti@gmail.com

## آنچه باید تحویل دهید

یک فایل zip. که حاوی پروژه نهایی شما است (آخرین نسخه از کدتان که کامیت کردید)

### نکات تحویل

- همچنین دقت شود برای ایجاد یک تجربه کاربری ساده و آسان، تمامی این خدمات باید با رابط کاربری کاربرپسند همراه باشند تا کاربران بتوانند به‌راحتی از هر خدمت بهره ببرند.
- دقت کنید که تمام مراحل این فاز پروژه به صورت گروهی دنبال شود و تمام اعضا در تحقیق و انتخاب موضوع نقش داشته باشند.
- فایل zip خواسته شده را سرگروه در سایت courses آپلود کند.
- پیاده‌سازی میکروسرویس بدون تنظیم و اجرای صحیح ساختار Docker و Gateway تیم، ناقص تلقی شده و به‌عنوان تحویل کامل این فاز در نظر گرفته نخواهد شد.
- فرمت نام فایل zip به صورت P7_GroupName.zip باشد که در آن GroupName نام گروه است.