به نام خدا

پروژه درس مهندسی نرم افزار – نیم‌سال اول ۱۴۰۴

عنوان پروژه: سیستم جامع انتخاب واحد

مدرس:مهندس امید ادیب‌فر

هدف: هدف این پروژه، پیاده‌سازی یک سیستم نرم‌افزاری کامل با استفاده از اصول و فرآیندهای مهندسی نرم‌افزار است. این پروژه صرفاً یک تمرین کدنویسی نیست؛ بلکه یک شبیه‌سازی از توسعه یک محصول واقعی در یک تیم چابک (Agile) است. شما باید نشان دهید که می‌توانید نیازمندی‌های یک سیستم را تحلیل کرده، معماری صحیحی (مبتنی بر اصول **SOLID**) طراحی کنید، آن را به صورت حرفه‌ای (با **Git**) مدیریت کرده، با **تست‌های واحد** کیفیت آن را تضمین نموده و در نهایت محصولی کارا تحویل دهید.

شرح پروژه:

یک پلتفرم مدرن، مبتنی بر API و تحت وب برای مدیریت و انجام فرآیند انتخاب واحد در دانشگاه است. این سیستم باید به سه گروه اصلی از کاربران خدمات دهد: **مدیران، اساتید** و **دانشجویان**.

نیازمندی های پروژه:

سیستم شما در نسخه نهایی باید حداقل قابلیت‌های زیر را داشته باشد:

نقش: مدیر (Admin):

- **احراز هویت:** مدیر باید بتواند با نام کاربری و رمز عبور وارد سیستم شود (Authentication).
- **مدیریت دروس (CRUD):** مدیر باید بتواند درس جدید تعریف کند، دروس موجود را ویرایش کند، لیست دروس را ببیند و درسی را حذف کند.

- اطلاعات هر درس شامل: نام درس، کد درس، ظرفیت، استاد مربوطه، زمان و مکان برگزاری (مثلاً: شنبه ۱۶-۱۴، کلاس ۳۰۱).

- **مدیریت پیش‌نیاز:** مدیر باید بتواند برای هر درس، یک یا چند درس دیگر را به عنوان «پیش‌نیاز» تعریف کند.
- **تعیین حد واحدهای اخذ شده:** مدیر بتواند حداقل و حداکثر واحدهای اخذ شده توسط یک دانشجو را مشخص کند. (فقط مخصوص دانشجویان آزمایشگاه)

نقش: دانشجو (Student)

- **احراز هویت:** دانشجو باید بتواند با شماره دانشجویی و رمز عبور وارد سیستم شود.
- **مشاهده دروس:** دانشجو باید بتواند لیست تمام دروس ارائه‌شده در ترم را ببیند.
- **جستجو و فیلتر:** دانشجو باید بتواند دروس را بر اساس نام درس یا نام استاد جستجو کند.
- **اخذ درس:** دانشجو باید بتواند یک درس را به لیست دروس اخذشده خود اضافه کند (ثبت‌نام کند).
- **مشاهده برنامه:** دانشجو باید بتواند برنامه هفتگی خود (شامل دروس اخذشده و زمان‌بندی آن‌ها) را به صورت بصری ببیند.
- **حذف واحد:** دانشجو باید بتواند یک درس را از لیست دروس اخذشده خود حذف کند. (فقط مخصوص دانشجویان آزمایشگاه)

نقش: استاد (Professor)

- **احراز هویت:** استاد باید بتواند با کد استادی و رمز عبور وارد سیستم شود.
- **مشاهده دروس:** استاد باید بتواند لیست تمام دروس ارائه‌شده مربوط به خود در ترم را ببیند.
- **مشاهده ثبت نام ها:** استاد باید بتواند لیست تمام دانشجویانی که در هر یک از دروس خود ثبت نام کرده اند را بصورت مرتب سازی شده بر اساس نام خانوادگی ببیند.
- **حذف دانشجو:** استاد باید بتواند دانشجو را از درس خود حذف کند. (فقط مخصوص دانشجویان آزمایشگاه)

قوانین کلیدی کسب‌وکار (Business Logic)

چالش اصلی شما در **بکند (Backend)** پیاده‌سازی صحیح این قوانین است. سیستم در هنگام «اخذ درس» توسط دانشجو *باید* موارد زیر را به صورت خودکار در سرور چک کند:

۱. **بررسی پیش‌نیاز:** آیا دانشجو تمام دروس پیش‌نیاز این درس را قبلاً پاس کرده است؟

۲. **بررسی تداخل زمانی:** آیا این درس با دروس دیگری که دانشجو در حال حاضر اخذ کرده، تداخل زمانی دارد؟

۳. **بررسی ظرفیت:** آیا ظرفیت کلاس پر نشده است؟

٤. **بررسی تکرار:** آیا دانشجو این درس را قبلاً در همین ترم اخذ نکرده است؟

٥. بررسی حد واحدهای اخذ شده (فقط مخصوص دانشجویان آزمایشگاه)

٦. بررسی حذف واحد فقط درصورتی که واحد در ترم جاری اخذ شده باشد (فقط مخصوص دانشجویان آزمایشگاه)

۷. بررسی حذف دانشجو از درس فقط در صورتی که دانشجو در ترم جاری در درس ثبت نام کرده باشد (فقط مخصوص دانشجویان آزمایشگاه)

اگر هر یک از این موارد نقض شود، API باید یک پیام خطای مشخص و معنادار به فرانت‌اند برگرداند.

الزامات فنی و فرآیندی

نحوه انجام پروژه به اندازه خود پروژه اهمیت دارد. رعایت موارد زیر اجباری و بخش بزرگی از نمره شماست.

۱. مدیریت فرآیند (Agile/Scrum)

- پروژه باید در **حداقل ۳ اسپرینت (Sprint)** دو یا سه هفته‌ای توسعه یابد.
- تیم‌ها باید از یک ابزار مدیریت پروژه (مانند **Trello**, **تسکولو** یا **میزیتو**) برای مدیریت **Backlog** خود استفاده کنند.

- در انتهای هر اسپرینت، تیم باید یک **دموی کوتاه** (**Sprint Review**) از قابلیت‌های تکمیل‌شده در آن اسپرینت را ارائه دهد.

۲. مدیریت کد (Git)

- استفاده از **Git** اجباری است. ریپازیتوری شما (در GitHub یا GitLab) باید در دسترس استاد باشد.
- **Commit History** شما بررسی خواهد شد. کامیت‌ها باید معنادار و منظم باشند.
- **استفاده از Pull Requestها و شاخه ها.** (فقط مخصوص دانشجویان آزمایشگاه)

۳. معماری و بکند (Backend/SOLID)

- بکند باید با هر تکنولوژی دلخواه (مانند **Python/Django/Flask**، Node.js، ...) پیاده‌سازی شود.
- معماری بکند باید اصول **SOLID** را رعایت کند. (به طور خاص، انتظار می‌رود **اصل مسئولیت واحد** (**SRP**) و **اصل وارونگی وابستگی** (**DIP**) با جداسازی واضح منطق کسب‌وکار (Services) از منطق دسترسی به داده (Repositories) به وضوح دیده شود).

۴. ارتباط (API)

- ارتباط بین فرانت‌اند و بکند *فقط و فقط* باید از طریق **REST API** (مبتنی بر JSON) باشد.
- تیم‌ها باید مستندات API خود را (حداقل در قالب یک فایل Markdown یا Postman Collection) ارائه دهند. استفاده از Swagger یا OpenAPI توضیه میشود.

۵. فرانت‌اند (Frontend)

- فرانت‌اند باید با هر فریم‌ورک مدرن (مانند **React**, **Vue**, **Angular**) پیاده‌سازی شود.
- فرانت‌اند *نباید* هیچ‌کدام از قوانین کلیدی کسب‌وکار (بخش ۴) را در خود پیاده‌سازی کند. تمام این چک‌ها باید در بکند انجام شود.

۶. تست‌نویسی (Testing)

- نوشتن **تست‌های واحد** (**Unit Tests**) برای بکند اجباری است.

- تمام «قوانین کسب‌وکار» (بخش ۴) باید به طور کامل توسط تست‌های واحد پوشش داده شوند. (مثلاً تستی که چک می‌کند آیا سیستم جلوی ثبت‌نام با تداخل زمانی را می‌گیرد یا نه).

موارد امتیازی (Bonus Points)

تیم‌هایی که فراتر از حداقل‌ها عمل کنند، می‌توانند نمره اضافه (تا سقف نمره کامل درس) دریافت کنند. موارد زیر شامل امتیاز ویژه خواهند بود:

۱. **کانتینرسازی (Containerization)**: ارائه پروژه به صورت داکرایز شده (ارائه Dockerfile و docker-compose.yml).

۲. **امنیت پیشرفته:** پیاده‌سازی کامل Authentication با استفاده از JWT و Refresh Tokens.
(فقط مخصوص دانشجویان آزمایشگاه)

۳. **تست E2E:** نوشتن حداقل چند تست End-to-End با ابزارهایی مانند Selenium یا Cypress برای فرانت‌اند.

٤. **استقرار (Deployment)**: مستقر کردن نسخه نهایی پروژه روی یک سرور (مانند Vercel, Heroku, ...).

موفق باشید/ادیب‌فر