تسک فنی BackEnd

سلام!
برای ارزیابی توانایی‌های فنی شما در طراحی و پیاده‌سازی سیستم‌های مدیریت اطلاعات کاربران، یک تسک عملی و کوتاه آماده کرده‌ایم. لطفاً با تمرکز بر روی معماری صحیح، امنیت داده‌ها و کیفیت کد، این تسک را در مدت مشخص‌شده انجام دهید.

هدف پروژه

ساخت یک اپلیکیشن تحت وب برای مدیریت و مشاهده اطلاعات کاربران و سفارشات با توجه به نقش‌های مختلف (مشتری و مدیر) با استفاده از **ASP.NET Core** و **Entity Framework**. در این سیستم، مشتریان فقط به اطلاعات خود دسترسی دارند و مدیران می‌توانند به اطلاعات تمامی مشتریان دسترسی پیدا کنند. سیستم باید شامل قابلیت‌هایی مانند مشاهده اطلاعات حساب، سفارشات و فاکتورها باشد.

مدل دامنه

1. **مشتری (حساب):** نمایانگر یک مشتری با فیلدهایی مانند CustomerID، FirstName، LastName، Email و WalletBalance و Role.
2. **سفارش:** نمایانگر سفارش‌های مشتری با فیلدهایی مانند OrderID، CustomerID، Product، Quantity، TotalAmount و OrderDate.
3. **فاکتور:** نمایانگر فاکتورها با فیلدهایی مانند InvoiceID، OrderID، Amount، DueDate و Status.

وظایف و قابلیت‌های سیستم

۱. نمایش اطلاعات حساب (برای مشتری و مدیر):

- مشتریان می‌توانند فقط اطلاعات حساب خود را مشاهده کنند.
- مدیران می‌توانند اطلاعات تمامی حساب‌ها را مشاهده کنند.

۲. نمایش لیست سفارشات (برای مشتری و مدیر)

- مشتریان می‌توانند فقط سفارشات خود را مشاهده کنند. این قابلیت لیستی از سفارشات انجام شده هر حساب را به صورت مرتب شده نزولی بر اساس زمان سفارش نمایش می دهد.
- مدیران می‌توانند سفارشات تمامی مشتریان را مشاهده کنند.

۳. قابلیت ایجاد فاکتور برای یک سفارش

توضیحات:

در این قابلیت، باید امکان ایجاد فاکتور برای هر سفارش وجود داشته باشد. زمانی که یک سفارش ثبت می‌شود، فاکتوری مرتبط با آن ایجاد می‌شود که شامل جزئیات مربوط به سفارش مانند مبلغ کل، تاریخ سررسید و وضعیت پرداخت است. هر فاکتور باید به یک سفارش خاص متصل باشد.

ورودی‌ها:

- OrderID: شناسه سفارش که برای آن باید فاکتور ایجاد شود.
- CustomerID: شناسه مشتری که سفارش مربوط به آن است.
- DueDate: تاریخ سررسید پرداخت فاکتور.

فرآیند ایجاد فاکتور:

1. سیستم ابتدا باید سفارش با شناسه OrderID را پیدا کند.
2. پس از یافتن سفارش، باید مبلغ کل سفارش (TotalAmount) را از اطلاعات سفارش استخراج کرده و فاکتور را برای آن سفارش ایجاد کند.
3. فاکتور جدید باید شامل موارد زیر باشد:
   - InvoiceID: شناسه منحصر به فرد فاکتور که به طور خودکار ایجاد می‌شود.
   - OrderID: شناسه سفارش که فاکتور به آن مربوط است.
   - Amount: مبلغ کل فاکتور (بر اساس TotalAmount سفارش).
   - DueDate: تاریخ سررسید پرداخت.
   - Status: وضعیت فاکتور که به طور پیش‌فرض باید "Pending" (در حال انتظار) باشد.

۴. قابلیت پرداخت فاکتور از طریق کیف پول

توضیحات:

مشتری می‌تواند یک فاکتور را با استفاده از موجودی کیف پول خود پرداخت کند. برای این کار، سیستم باید بررسی کند که آیا موجودی کیف پول مشتری به اندازه مبلغ فاکتور کافی است یا خیر. در صورت کافی بودن موجودی، مبلغ از کیف پول کسر شده و وضعیت فاکتور به "پرداخت شده" تغییر می‌کند. در صورتی که موجودی کافی نباشد، پیغام خطای مناسب به مشتری نمایش داده می‌شود. در نظر داشته باشید که هر کاربر صرفا می‌تواند فاکتورهای خودش را پرداخت کند.

ورودی‌ها:

- InvoiceID: شناسه فاکتور که مشتری قصد پرداخت آن را دارد.

- CustomerID: شناسه مشتری که می‌خواهد فاکتور را پرداخت کند.

نکات:
- بررسی داده‌های ورودی: تمام ورودی‌ها باید بررسی شوند و در صورت نامعتبر بودن، خطای مناسب بازگشت داده شود.
- به‌روزرسانی وضعیت فاکتور: پس از پرداخت موفق، وضعیت فاکتور باید به "پرداخت شده" تغییر یابد.
- کاهش موجودی کیف پول: پس از پرداخت، موجودی کیف پول مشتری باید به‌روز شود.

۵. مشاهده فاکتورها (برای مشتری و مدیر)
- مشتریان فقط می‌توانند فاکتورهای خود را مشاهده کنند. این قابلیت لیستی از فاکتورهای هر حساب را به صورت مرتب شده نزولی بر اساس زمان DueDate نمایش می دهد.
- مدیران می‌توانند فاکتورها را برای تمامی مشتریان مشاهده کنند.

قابلیت‌های اختیاری (امتیاز اضافی)

- **قابلیت ثبت و پیگیری تراکنش‌های کیف پول برای هر کاربر** (در این قابلیت، باید امکان ثبت و پیگیری تمامی تراکنش‌های مربوط به کیف پول هر کاربر فراهم شود. این تراکنش‌ها شامل تمام واریزها و برداشت‌ها از کیف پول کاربر می‌باشد.)

- **Unit Testing:** باید از تست‌های واحد برای ارزیابی عملکرد و صحت کد استفاده کنید. این تست‌ها باید به‌طور خودکار انجام شوند و شامل تمامی بخش‌های مختلف سیستم باشند.

نکات مهم در پیاده‌سازی

- از **احراز هویت مبتنی بر JWT** برای شناسایی کاربران استفاده کنید.
- **مدیریت خطاها و پیام‌های مناسب:** به‌طور ویژه به مدیریت صحیح خطاها و پیام‌های سیستم توجه کنید. سیستم باید پیام‌های خطای مناسب و دقیق بر اساس نوع مشکل (برای مثال، ورودی نامعتبر یا دسترسی غیرمجاز) ارائه دهد.
- **ساختار کدنویسی تمیز:** بسیار مهم است که کدها به‌طور منظم، خوانا و قابل نگهداری نوشته شوند. از اصول SOLID و الگوهای طراحی استاندارد برای بهبود قابلیت توسعه و تست کد استفاده کنید. همچنین، متغیرها و توابع باید اسامی معنادار و دقیقی داشته باشند.

نحوه ارسال پروژه

- سورس‌کد را در GitHub آپلود کرده و لینک آن را برای ما ارسال کنید.
- توضیحات لازم و تصمیمات فنی را در فایل README اضافه کنید.

محدودیت زمانی

ما توقع داریم که بیشتر از **۶ ساعت** برای انجام این تسک وقت نگذارید. ما بیشتر به روش تفکر شما، ساختار کد و تجربه کاربری اهمیت می‌دهیم تا ظاهر نهایی یا حجم کد.

موفق باشید!