# بررسی کد، یافته‌ها و اصلاحات انجام‌شده

در این مرحله، نمونه اولیه توسعه‌یافته سامانه آموزش آنلاین تحت فرآیند Code Review قرار گرفت تا علاوه بر شناسایی خطاهای احتمالی، میزان انطباق پیاده‌سازی با معماری طراحی‌شده در فازهای قبل، اصول شیءگرایی، الگوهای طراحی GoF و اصول SOLID ارزیابی شود. بررسی کد صرفاً محدود به یافتن خطاهای نحوی نبود، بلکه ساختار وابستگی‌ها، مسئولیت کلاس‌ها، کیفیت طراحی، قابلیت توسعه، خوانایی و قابلیت نگهداری سیستم نیز مورد تحلیل قرار گرفت. تمامی یافته‌ها پس از بررسی، اصلاح شده و نسخه نهایی پروژه بر اساس آن‌ها بازنگری گردید.

## روش انجام Code Review

فرآیند بازبینی کد بر اساس معیارهای زیر انجام شد:

- انطباق پیاده‌سازی با مستندات طراحی فازهای چهارم و پنجم
- رعایت اصول SOLID
- رعایت مسئولیت یگانه (Single Responsibility)
- بررسی وابستگی بین لایه‌های معماری
- بررسی نحوه پیاده‌سازی Design Patternهای انتخاب‌شده
- بررسی کیفیت نام‌گذاری کلاس‌ها و متدها
- بررسی جریان داده بین Repository، Service، Controller و Domain
- بررسی قابلیت توسعه و نگهداری سیستم

## یافته‌های اصلی Code Review

### یافته اول: جداسازی مناسب لایه‌های معماری

بررسی ساختار پروژه نشان داد که منطق تجاری، مدیریت داده، کنترل درخواست‌ها و مدل‌های دامنه در لایه‌های مستقل پیاده‌سازی شده‌اند. این جداسازی موجب کاهش وابستگی بین اجزاء و افزایش قابلیت نگهداری سیستم شده است.

**وضعیت:** اصلاح نیاز نداشت.

### یافته دوم: پیاده‌سازی صحیح Repository Pattern

تمام عملیات دسترسی به داده از طریق Interfaceهای Repository انجام شده و هیچ‌یک از سرویس‌ها وابستگی مستقیمی به ساختار ذخیره‌سازی اطلاعات ندارند. این موضوع امکان جایگزینی Repositoryهای حافظه‌ای با پایگاه‌داده واقعی را در آینده فراهم می‌کند.

**اصلاح انجام‌شده:**

در نسخه اولیه، اطلاعات مربوط به Submission به‌صورت محلی در Service مدیریت می‌شد. در نسخه نهایی، مخزن مستقل SubmissionRepository و Interface متناظر آن اضافه شد تا مدیریت داده مطابق Repository Pattern انجام گیرد.

## یافته سوم: بهبود مدیریت موجودیت‌های دامنه

تمام موجودیت‌های اصلی سیستم از کلاس BaseEntity ارث‌بری می‌کنند و شناسه یکتای مبتنی بر UUID دریافت می‌کنند. این موضوع یکنواختی مدل دامنه و جلوگیری از تکرار کد را فراهم کرده است.

**اصلاح انجام‌شده:**

کلیدهای Repository از رشته به شیء UUID تغییر یافتند تا از تبدیل‌های غیرضروری جلوگیری شده و یکپارچگی داده افزایش یابد.

## یافته چهارم: بهبود فرآیند نمره‌دهی

الگوی Strategy به‌درستی برای انتخاب روش تصحیح آزمون پیاده‌سازی شده است و Decorator بدون تغییر در الگوریتم اصلی، جریمه تأخیر را اعمال می‌کند.

**اصلاح انجام‌شده:**

فرمول جریمه تأخیر بازنگری شد و به جای کسر مقدار ثابت، کاهش درصدی نمره اعمال گردید. این اصلاح باعث شد منطق محاسبه نمره مستقل از مقدار نهایی آزمون باقی بماند.

## یافته پنجم: اصلاح منطق قبولی آزمون

در نسخه اولیه، وضعیت قبولی بر اساس مقدار ثابت ارزیابی می‌شد که برای آزمون‌های با نمره کل متفاوت مناسب نبود.

**اصلاح انجام‌شده:**

معیار قبولی به صورت درصدی از نمره کل آزمون محاسبه شد و مقدار آستانه قبولی از طریق پارامتر passThreshold قابل تنظیم گردید.

## یافته ششم: کاهش وابستگی بین اجزای سیستم

تمام سرویس‌ها وابستگی‌های موردنیاز خود را از طریق Constructor دریافت می‌کنند و هیچ نمونه‌سازی مستقیمی در داخل کلاس‌ها انجام نمی‌شود.

**اصلاح انجام‌شده:**

Dependency Injection در تمامی Serviceها بازنگری شد تا وابستگی‌ها تنها از بیرون تزریق شوند.

### یافته هفتم: بهبود ساختار Controller

در نسخه نهایی، تمامی عملیات سطح کاربرد در کلاس AppController متمرکز شد و این کلاس صرفاً نقش هماهنگ‌کننده بین لایه ارائه و سرویس‌ها را بر عهده دارد. منطق تجاری همچنان در Service Layer باقی مانده است.

**اصلاح انجام‌شده:**

تفکیک کامل مسئولیت بین Controller و Service حفظ شد و از انتقال منطق تجاری به Controller جلوگیری گردید.

### یافته هشتم: بازنگری ماژول Event

ساختار Publisher و Listenerها بررسی شد تا وابستگی مستقیم بین تولیدکننده و مصرف‌کننده رویدادها حذف گردد.

**اصلاح انجام‌شده:**

ماژول Event به صورت مستقل سازمان‌دهی شد و کلاس‌های Listener برای پردازش رویدادهای اعلان و آزمون تفکیک گردیدند تا امکان توسعه رویدادهای جدید بدون تغییر در Publisher فراهم شود.

## خلاصه اصلاحات انجام‌شده

| ردیف | یافته | اصلاح انجام‌شده | نتیجه |
|---|---|---|---|
| 1 | مدیریت Submission در Service | ایجاد `ISubmissionRepository` و `SubmissionRepository` | افزایش انطباق با Repository Pattern |
| 2 | استفاده از کلیدهای متنی | استفاده مستقیم از UUID | افزایش یکپارچگی داده |
| 3 | جریمه تأخیر | اصلاح فرمول به جریمه درصدی | افزایش دقت محاسبات |
| 4 | منطق قبولی | محاسبه بر اساس درصد نمره کل | افزایش انعطاف‌پذیری سیستم |
| 5 | تزریق وابستگی‌ها | بازنگری Constructorها | رعایت Dependency Injection |
| 6 | Controller | تمرکز عملیات در AppController بدون انتقال منطق تجاری | حفظ معماری لایه‌ای |
| 7 | Event Module | تفکیک Publisher و Listenerها | کاهش وابستگی و افزایش توسعه‌پذیری |
| 8 | Repositoryها | بازنگری ساختار Interfaceها | افزایش قابلیت توسعه |

## جمع‌بندی

نتایج بازبینی نشان داد که نسخه نهایی سامانه آموزش آنلاین از نظر ساختار معماری، رعایت اصول طراحی شیءگرا و الگوهای طراحی در وضعیت مطلوبی قرار دارد. اصلاحات انجام‌شده موجب افزایش خوانایی کد، کاهش وابستگی بین اجزا، بهبود قابلیت توسعه، افزایش قابلیت نگهداری و انطباق بیشتر پیاده‌سازی با مستندات طراحی فازهای پیشین شده است. همچنین استفاده از Repository Pattern، Strategy Pattern، Decorator Pattern، Dependency Injection و معماری لایه‌ای سبب شده است که سیستم برای توسعه قابلیت‌های آتی مانند اتصال به پایگاه‌داده واقعی، افزودن سرویس‌های جدید یا تغییر سیاست‌های نمره‌دهی، بدون نیاز به بازطراحی هسته سامانه آماده باشد. این نتایج نشان می‌دهد که فرآیند Code Review صرفاً به اصلاح خطاها محدود نبوده، بلکه نقش مؤثری در ارتقای کیفیت طراحی و افزایش پایداری نرم‌افزار ایفا کرده است.