Humayun 03-06-2019

Assignment description:
https://github.com/chamatheapp/chama-backend-assignment-course-signup

Candidate profile: https://www.linkedin.com/in/humayunahmed/

Applicant's result:

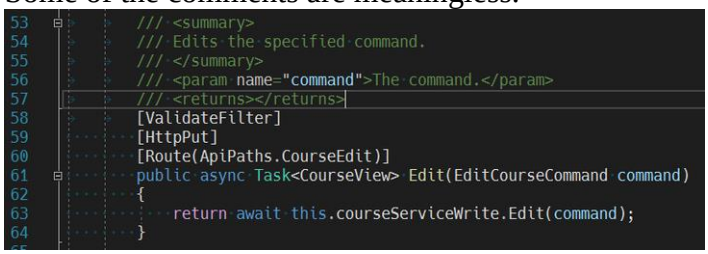
I have written 2 applications to support the requirements. The technical requirement was really big, specially the advanced version. Due to the time limitation, I have not completed the presentation yet, I'll do and update the github repository. But basic documentation I have given there. Please check out following 2 repositories:
1. https://github.com/humayun-ahmed/OnlineCourse.git
2. https://github.com/humayun-ahmed/CQRS.git
For documentation (which I'll update more), please check the following url:https://github.com/humayun-ahmed/OnlineCourse/blob/master/Project%20Overview.docx

Let me know if you have any questions.

Regards,
Humayun

	Comments
• Writing testable code	▪ Although the code seems testable I didn't see any valuable test that asserts the business requirements. ▪ Used the anti-pattern Service Locator, making it really difficult to determine what are the dependencies in the class (see IDependencyResolver being injected in the class CourseServiceWrite). This compromises the testability of the system. ▪ The basic elements are testable
• Test infrastructure	▪ Cannot determine from what I could see. ▪ I saw usage of Moq and a builder called FizzWare, which I don't think is widely used.
• Exception handling and logging	▪ There was no clear error handling and no expressive logging. ▪ Used NLog and some commented code for integration with Azure AI.

• Asynchronous code (when and how (not) to use async/await)	▪ Awating tasks when not needed.
• Asynchrony through messaging (events, commands)	
• Knowledge of Azure infrastructure and storage technologies platform (WebJobs, Azure Functions,SQL, Table Storage, Cosmos DB, ...)	• No evidence of usage (but this doesn't mean he doesn't understand it).
• Code organisation (modularity, dependencies between modules, etc)	▪ Too many files in the repo ▪ Really hard to tell what is code from the candidate and what is bloatware (see Infrastructure folder, for example) ▪ Far too many projects, far too much YAGNI for a simple requirement. ▪ DI - Property Injection considered harmful. 🙂
• Namings	
• Domain model design (OOP, DDD concepts, etc)	▪ Anemic domain model ▪ Saw DDD concepts on naming but no real value (see BaseAggregateRoot, IAggregate are empty) ▪ Seems to show understanding of commands vs events.
• Handling concurrency (avoid exceeding the course capacity)	▪ There was a check that did validate the amount of students when signing up students, but it doesn't really solve the concurrency issue.
Overall Feeling	
▪ Assignment completion	• One solution did not compile after first clone.

▪ Overall code quality (edge cases, usage of tools)	• A real mixture of styles, formatting, third-party code. Mix of tabs/spaces, line spacing. Lots of code left commented. Very difficult to read. • Some of the comments are meaningless. 

```
53      /// <summary>
54      /// Edits the specified command.
55      /// </summary>
56      /// <param name="command">The command.</param>
57      /// <returns></returns>
58      [ValidateFilter]
59      [HttpPut]
60      [Route(ApiPaths.CourseEdit)]
61      public async Task<CourseView> Edit(EditCourseCommand command)
62      {
63          return await this.courseServiceWrite.Edit(command);
64      }
```