- Part I. System architecture and system design (24.5/30 marks total)
  - Part I.1. (3.5/5 marks) - (After a “write operation”, “read operation” will see it. should be written to -> "Strong Consistency ensures that after a 'write operation,' a subsequent 'read operation' will see the most recent write) & No specific use cases where eventuall consistent persistences are acceptable
  - Part I.2. (5/5 marks)
  - Part I.3. (3/5 marks) - No explain the differences between point-to-point and pub/sub patterns & AWS Copilot as AWS architecture references for pub/sub is irrelevant due to AWS Copilot is just a CLI tool itself
  - Part I.4. (5/5 marks)
  - Part I.5. (5/5 marks)
  - Part I.6. (3/5 marks) - Expects listing trade off other than adding complexity
- Part II. Classic AWS Architecture design (26/30 marks total)
  - Part II.1. - Base design questions (18/20 marks) - For Security Group, missing allow inbound, outbound port range for EFS, for Route Table -> VPC Endpoint can't attach with Route Table
  - Part II.2. - Expansive network design questions (5/10 marks) - Not use S3/DynamoDB VPC endpoint, for the Orange flow, the API Service must route request to NAT GW (not the Load Balancer)
  - Part II.3. - Bonus design questions (bonus 3/5 marks) - Should use AppMesh and CloudMap instead of Route53
- Part III. AWS Native Architecture design (6/10 marks total) - For website serving, has no Dynamic data serving (API GW & Lambda & DynamoDB), for scheduled update, has no event scheduler to trigger the lambda (I suggest EventBridge Scheduler), good improvement with Step Functions and 2 dynamodb tables to track crawler history
- Part IV. AWS Native Software Development (31/30 marks total)
  - Part IV.1. Setup source events (10/10 marks)
  - Part IV.2. Checking in (10/10 marks)
  - Part IV.3. Making a claim (10/10 marks)
  - Part IV.4. Bonus mark for quality (1/5 bonus mark)