|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | შეუძლია ძირითადი ამოცანების შესრულება | მუდმივად აქვს მაღალი ხარისხის სტანდარტების შესაბამისი შედეგები | წარმატებული ექსპერტი თავის სფეროში | აღიარებულია, როგორც ტალანტი თავის სფეროში | გარე დონეზე აღიარებულია, როგორც მსხვილი მასშტაბების ექსპერტი | წონა | წონა % |
| **ცოდნა და უნარ-ჩვევები** | **1** | **2** | **3** | **4** | **5** |  |  |
| ობიექტზე ორიენტირებული  პროგრამირება | აქვს ობიექტზე ორიენტირებული პროგრამირების ერთი ან მეტი ენის საბაზისო ცოდნა. ძირითადად დამოუკიდებლად შეუძლია ამ ენებზე კოდის წერა ან არსებულ კოდში ცვლილებების შეტანა. | კარგად აქვს გააზრებული OOP-ის Encapsulation და Abstraction პრინციპები, იცის მათი რეალიზების ტექნიკები და ოპტიმალურად აკეთებს კლასის ინტერფეისის დიზაინს, უზრუნველყოფს მისი შიდა იმპლემენტაციის დამოუკიდებლობას, იზოლაციას კლიენტ კოდისგან. | კარგად აქვს გააზრებული OOP-ის Composition, Inheritance და Delegation პრინციპები, მათ შორის სხვაობა, იცის მათი რეალიზების ტექნიკები და ოპტიმალურად აკეთებს კომპლექსური კლასების დიზაინს, იმპლემენტაციას, ობიექტების reuse-ს.  კარგად აქვს გააზრებული OOP-ის Polymorphism პრინციპი, მისი რეალიზების ტექნიკები და ოპტიმალურად აკეთებს მსგავსი ფუნქციის, მაგრამ განსხვავებული იმპლემენტაციის მქონე კლასებისთვის საერთო ინტერფეისის დიზაინს და უზრუნველყოფს კლიენტ კოდისთვის მათ მარტივ გამოყენებას, ურთიერთჩანაცვლებას. | კარგად აქვს გააზრებული Single Responsibility (SRP) და  Interface Segregation (ISP) პრინციპები, იცის მათი რეალიზების ტექნიკები და  კლასების დიზაინს აკეთებს  კონკრეტული, მჭიდროდ დაკავშირებული ფუნქციონალის რეალიზების შესაბამისად (Only one reason to change), ხოლო ინტერფეისების დიზაინს აკეთებს კლიენტ კოდის საჭიროების შესაბამისად. | კარგად აქვს გააზრებული Open/Closed (OCP) და Liskov Substitution (LSP) პრინციპები, იცის მათი რეალიზების ტექნიკები და კლასების დიზაინის დროს ითვალისწინებს მათი გაფართოების/განვრცობის (Extension) შესაძლებლობას,  მშობელი კლასის ობიექტის ნაცვლად განვრცობილი კლასის ობიექტის უპრობლემოდ გამოყენების შესაძლებლობას.  კარგად აქვს გააზრებული Dependency Inversion (DIP) პრინციპი და  უზრუნველყოფს, რომ ზედა დონის კლასი (ბიზნეს ლოგიკის კლასი) არ იყოს დამოკიდებული ქვედა დონის კლასზე (ტექნიკური/ინფრასტრუქტურული კლასი), არამედ ორივე დამოკიდებული იყოს აბსტრაქციებზე. ამასთან, აბსტრაქციები არ იყოს დამოკიდებული იმპლემენტაციის დეტალებზე, არამედ იმპლემენტაციის დეტალები იყოს დამოკიდებული აბსტრაქციებზე. | 10 | 9% |
| რელაციური მონაცემთა ბაზები | აქვს ერთი ან მეტი რელაციური მონაცემთა ბაზის საბაზისო ცოდნა (SQL syntax, Data types, …).  შეუძლია დამოუკიდებლად ძირითადი ობიექტების შექმნა (Schemas, Tables, Views, ...), მათ შორის რელაციების აწყობა, მონაცემების ჩაწერა, განახლება, წაკითხვა და წაშლა. | კარგად იცნობს ბაზის მოდელირების და ნორმალიზაციის ტექნიკებს (1NF, 2NF, 3NF), შესაბამისად აპროექტებს მონაცემთა ბაზის ობიექტებს.  აქვს ბაზის პროცედურული ენ(ებ)ის (TSQL, PL/SQL, …) საბაზისო ცოდნა.  აქვს ტრანზაქციულობის და მონაცემთა მდგრადობის, მონაცემების საიმედოდ შენახვის უზრუნველყოფის ტექნიკების ცოდნა.  დამოუკიდებლად შეუძლია ამოცანის შესრულება ან არსებულ ამოცანაში ცვლილებების შეტანა. | აქვს ბაზის პროცედურული ენ(ებ)ის (TSQL, PL/SQL, …) კარგი ცოდნა.  აქვს Concurrency მოდელების/მეთოდების, იზოლაციის დონეების კარგი ცოდნა.  შეუძლია Query profiling და Execution plan-ის ანალიზი, რათა შექმნას ოპტიმალური ჩაწერა-წაკითხვის ოპერაციები.  დამოუკიდებლად შეუძლია, რთული ამოცანის შესრულება.  აქვს ETL-ების და შესაბამისი ინსტრუმენტ(ებ)ის საბაზისო ცოდნა და შეუძლია ამოცანის შესრულება. | აქვს ინდექსის ტიპების, სტრუქტურის და არქიტექტურის კარგი ცოდნა.  შეუძლია ოპტიმალური ინდექსების შექმნა, რათა უზრუნველყოს მონაცემების სწრაფი წაკითხვა, მონაცემების ჩაწერის წარმადობის დეგრადაციის გარეშე.  აქვს ETL-ების და შესაბამისი ინსტრუმენტ(ებ)ის კარგი ცოდნა და შეუძლია რთული ამოცანის შესრულება.  აქვს დიდი მონაცემების ჩაწერა-წაკითხვის ოპტიმალური ტექნიკების ცოდნა (Bulk operations) და შეუძლია შესაბამისი რთული ამოცანის შესრულება. | აქვს მონაცემთა ბაზის წარმადობის ანალიზის, მეტრიკების და დიაგნოსტირების Advanced ტექნიკების ცოდნა.  მათზე დაყრდნობით შეუძლია ბაზის ოპტიმიზაცია და აქვს შესაბამისი ოპტიმიზაციის Advanced ტექნიკების ცოდნა.  აქვს დიდი რაოდენობის მონაცემების მქონე ბაზების ოპტიმიზაციის ტექნიკების კარგი ცოდნა, როგორიცაა Data Retention & Archiving, Database Table Partitioning. იყენებს ამ ტექნიკებს მონაცემთა ბაზების ოპტიმიზაციისთვის. | 8 | 7% |
| ინტეგრაციები, სერვისები და მესიჯინგი | აქვს ვებ სერვისების და API-ების, SOAP და REST პროტოკოლების საბაზისო ცოდნა. შეუძლია დამოუკიდებლად შეასრულოს ამოცანა ორივე ტიპის სერვისების გამოძახებით, მონაცემების მისაღებად ან ოპერაციის შესასრულებლად. | აქვს REST და SOAP პროტოკოლის ძირითადი საკითხების ცოდნა და შეუძლია სერვისების შექმნა დამოუკიდებლად.  იცნობს სერვისების შემუშავების Good API კრიტერიუმებს და ამის შესაბამისად ქმნის სერვისებს.  იცნობს API Gateway პლატფორმას და შეუძლია სერვისის გამოქვეყნება ამ პლატფორმაზე, ასევე სხვა გამოქვეყნებული სერვისების გამოყენება.  აქვს ერთი ან მეტი მესიჯინგ სისტემის (მესიჯ ბროკერის - RabbitMQ, EMS) საბაზისო ცოდნა და შეუძლია მესიჯების საიმედოდ გამოქვეყნება, მიღება და დამუშავება. | აქვს REST პროტოკოლის კარგი ცოდნა: Resource Naming and Routing, Message format and structure, HTTP Verbs/Methods, HTTP Status Codes, Error handling, ...  აქვს SOAP პროტოკოლის კარგი ცოდნა: WSDL and XSD schemas, Message format and structure, SOAP Envelope, Transport types, Error handling, ...  აქვს ერთი ან მეტი მესიჯინგ სისტემის (მესიჯ ბროკერის - RabbitMQ, EMS) და შესაბამისი პროტოკოლის (AMQP, JMS) კარგი ცოდნა. შეუძლია დამოუკიდებლად მესიჯინგ არხების და ტოპოლოგიის ოპტიმალური დიზაინი, შექმნა და გამოყენება. | აქვს სინქრონული და ასინქრონული კომუნიკაციის პატერნების კარგი ცოდნა (RPC, Request-Response, Request-Reaction, Publish-Subscribe, Inbox, Outbox) და იყენებს მათ ამოცანის ოპტიმალურად და საიმედოდ შესასრულებლად.  სერვისის შემუშავებისას ითვალისწინებს შემდეგ საკითხებს: Versioning, Idempotence, Statelessness, Resilience (Timeout, Retry, Circuit Breaker), Compression, Caching. | აქვს სერვისების და მესიჯინგის დიზაინის და იმპლემენტაციის Advanced საკითხების ცოდნა, უზრუნველყოფს მათ მაღალ მდგრადობას, მასშტაბირებას და ხელმისაწვდომობას (Advanced Message Routing, Clustering, Sharding, Fault Tolerance, Consistent Hashing, Monitoring and Diagnostics). | 7 | 6% |
| კოდის ორგანიზება,  დიზაინ პატერნები | აქვს კოდის ორგანიზების ძირითადი პრინციპების ცოდნა, როგორიცაა DRY, YAGNI, KISS და მათ შესაბამისად წერს კოდს. | დამოუკიდებლად შეუძლია პროექტის კოდის ორგანიზება აპლიკაციის სხვადასხვა ასპექტების ერთმანეთისგან გამიჯვნით (Separation of Concerns), მარტივად და ცხადად აღსაქმელი სტრუქტურების და მათი დამოკიდებულებების შექმნით.  კარგად იცნობს ლეგიონის და არეას სტანდარტებს და გაიდლაინებს, მათ შესაბამისად აკეთებს პროექტის კოდის ორგანიზებას. | აქვს დიზაინ პატერნების კარგი ცოდნა, როგორიცაა Creational, Structural, Behavioral პატერნები და იყენებს მათ კოდის წერის დროს. | კარგად აქვს გააზრებული Component პრინციპები,  Component cohesion principles:   * REP: The Reuse/Release Equivalence Principle * CCP: The Common Closure Principle * CRP: The Common Reuse Principle   Component coupling principles:   * ADP: Acyclic Dependencies Principle * SDP: Stable Dependencies Principle * SAP: Stable Abstractions Principle   და ამ პრინციპების შესაბამისად წერს კოდს, ახდენს არსებული კოდის ოპტიმიზაციას. | პროექტის კოდის ორგანიზებისას იყენებს ცნობილ პრინციპებს, პატერნებს და პრაქტიკებს როგორებიცაა Clean architecture, Hexagonal architecture, Onion architecture და სხვა. | 6 | 5% |
| სისტემის პროექტირება,  არქიტექტურული პატერნები | დამოუკიდებლად შეუძლია არსებული სისტემის არქიტექტურის გარჩევა და მის დეველოპმენტში კონტრიბუციის კეთება, არჩეული არქიტექტურის პრინციპების შესაბამისად. | დამოუკიდებლად შეუძლია 3-tier (Presentation – Application - Database) აპლიკაციის ოპტიმალური დაპროექტება და აწყობა.  სისტემის კომპონენტებს შორის კომუნიკაციისა და ინტეგრაციისთვის იყენებს ოპტიმალურ კომუნიკაციის/ინტეგრაციის პატერნებს, პროტოკოლებს და ინფრასტრუქტურას.  კარგად იცნობს ბანკის არქიტექტურის პრინციპებს, ლეგიონის და არეას სტანდარტებს და გაიდლაინებს, მათ შესაბამისად აკეთებს სისტემის არქიტექტურას. | დამოუკიდებლად შეუძლია Multi-tier აპლიკაციის ოპტიმალური დაპროექტება.  აქვს სერვისზე ორიენტირებული არქიტექტურის (SOA) პრინციპების და პრაქტიკების კარგი ცოდნა, მათ შესაბამისად აკეთებს სისტემის დიზაინს და იმპლემენტაციას.  აქვს Domain Driven Design-ის სტრატეგიული და ტაქტიკური პატერნების კარგი ცოდნა და იყენებს მათ სისტემის დიზაინის დროს. | აქვს Microservices არქიტექტურის პრინციპების და პრაქტიკების კარგი ცოდნა, მათ შესაბამისად აკეთებს მიკროსერვისების დიზაინს და იმპლემენტაციას.  იყენებს CQRS პატერნს და ოპტიმალურად აკეთებს ჩაწერის (Command) და წაკითხვის (Query) ოპერაციებს, მათი ერთმანეთისგან იზოლაციით და კონკრეტულ ასპექტზე მორგებული მოდელებით, მონაცემთა ბაზით თუ სხვა ინფრასტრუქტურით.  იყენებს Saga პატერნს და უზრუნველყოფს დისტრიბუციულ სისტემაში განაწილებული პროცესების ოპტიმალურ მართვას და ტრანზაქციული მთლიანობის შენარჩუნებას (Distributed consensus). | აქვს Event-driven არქიტექტურის კარგი ცოდნა, მათ შესაბამისად აკეთებს სისტემის დიზაინს და იმპლემენტაციას.  იყენებს Event Sourcing პატერნს და უზრუნველყოფს ბიზნეს ობიექტის ცვლილების ყველა Event-ის შენახვას და დისტრიბუციას, ობიექტის მდგომარეობის წარმოდგენას Event-ების სახით, ასევე ობიექტის ისტორიული მდგომარეობის აღდგენას (replay) და მისი პროექციების შექმნას.  აქვს Actor model პატერნის და პრაქტიკების კარგი ცოდნა, მათი გამოყენებით შეუძლია დისტრიბუციულ სისტემაში შესასრულებელი ამოცანების ოპტიმალური განაწილება და მასშტაბირება. | 6 | 5% |
| Unit და Integration ტესტები | აქვს Unit და Integration ტესტების საფუძვლების (AAA) და ფრეიმვორკ(ებ)ის ცოდნა. დამოუკიდებლად შეუძლია Unit და Integration ტესტების გარჩევა და დაწერა. | კარგად იცნობს და იყენებს კოდის პროექტის ყველა შრისთვის (Layer) Unit და Integration ტესტების დასაწერად საჭირო მიდგომებს და ტექნიკებს, როგორიცაა Setup and Teardown, Parameterizing, Mocks, Fakes, Assertions, Shared Contexts and Fixtures, TestServer, WebApplicationFactory.  იცნობს და იყენებს კოდის ტესტებით დაფარვის მაჩვენებლის გაზომვის ინსტრუმენტებს. | დეველოპმენტის დროს Unit და Integration ტესტების დაფარვის მაჩვენებელი აქვს 80% ან მეტი.  დამოკიდებული სერვისების და API-ების გასატესტად იყენებს API Mocking ინსტრუმენტებს. | აქვს Test Driven Development-ის ცოდნა და შესაბამისად წერს ტესტებს. | აქვს Behavior Driven Development-ის ცოდნა და შესაბამისად წერს ტესტებს. | 7 | 6% |
| კოდის ვერსიების კონტროლი და  კოლაბორაცია | აქვს სორს კოდის ვერსიების კონტროლის სისტემ(ებ)ის ცოდნა და მუშაობის გამოცდილება, როგორიცაა Git, SVN. დამოუკიდებლად შეუძლია კოდის განთავსება, წამოღება, branch-ის შექმნა და და-merge-ვა. | აქვს branching-ის სტრატეგიების ცოდნა და იყენებს პრაქტიკაში, როგორიცაა release flow, trunk based development, git flow. | ერთობლივი კოდის წერის დროს იყენებს pull request და code review ტექნიკებს. | აქვს სორს კონტროლის სისტემაში branch policy-ების, კონვენციების და კოდის ავტომატური ვალიდაციების კონფიგურაციის ცოდნა და გამოცდილება. | აქვს ვერსიების კონტროლის Advanced ტექნიკების ცოდნა და გამოცდილება, როგორებიცაა: Rewriting History, Hooks, Repository Split, Searching, … | 8 | 7% |
| DevOps, CI/CD | N/A | იცნობს DevOps-ის კონცეფციას და დეველოპმენტის დროს ითვალისწინებს მას, უზრუნველყოფს ავტომატიზაციის მიმართ პროექტის მზაობას.  აქვს ავტომატური build-ის და დანერგვის საშუალებების გამოყენების ცოდნა და გამოცდილება, როგორებიცაა TFS/Azure Devops, Jenkins ან სხვ. | იცნობს ბანკში არსებულ DevOps და CI/CD პლატფორმებს, მათ სტანდარტულ ფუნქციონალს და შეუძლია ამ პლატფორმების გამოყენებით პროექტის სტანდარტული Pipeline-ის აწყობა. | დეტალურად იცნობს ბანკში არსებულ DevOps და CI/CD პლატფორმების შესაძლებლობებს, იყენებს მათ პროექტების ავტომატიზაციაში. | მუდმივად ეცნობა DevOps და CI/CD საკითხებთან დაკავშირებულ სიახლეებს და ეხმარება გუნდს ახალი ავტომატიზაციის საშუალებების შემოტანასა და დანერგვაში. | 2 | 2% |
| სისტემის უსაფრთხოება,  უსაფრთხო დეველოპმენტი | ძირითადად დამოუკიდებლად შეუძლია უსაფრთხო კოდის წერა და იცნობს დეველოპმენტში გამოყენებული ფრეიმვორკის (ASP.Net, Angular, ...) უსაფრთხოებასთან დაკავშირებულ მოთხოვნებს/თავისებურებებს. | აქვს უსაფრთხო კოდის წერის ცოდნა და გამოცდილება ისეთი საფრთხეების გათვალისწინებით როგორიცაა: Injection, Broken Authentication, XSS, CSRF.  კარგად იცნობს ბანკის ინფორმაციული უსაფრთხოების წესებს, პოლიტიკებს და გაიდლაინებს. მათ შესაბამისად აკეთებს დეველოპმენტს.  აკეთებს სტატიკური კოდის ანალიზს და დამოკიდებული ბიბლიოთეკების გადამოწმებას ბანკში დანერგილი ინსტრუმენტების შესაბამისად (Fortify SCA, OWAS Dependency-Check). | კარგად იცის OWASP Top 10 საფრთხეები და მათი გათვალისწინებით წერს უსაფრთხო კოდს, აკეთებს არსებული აპლიკაციების ოპტიმიზაციას, უსაფრთხოების პრობლემების აღმოფხვრას.  აქვს აუტენტიფიკაციის, ავტორიზაციის და სესიების უსაფრთხოდ მართვის ძირითადი საკითხები ცოდნა და გამოცდილება. | აქვს აუტენტიფიკაციის და ავტორიზაციის პროტოკოლების და შესაბამისი პროცესების კარგი ცოდნა და იმპლემენტაციის გამოცდილება, როგორებიცაა: OpenID Connect, OAuth 2, ASP.NET (Core) Identity. | აქვს სისტემების შეღწევადობის (პენეტრაციის) ტესტების ჩატარების ცოდნა და გამოცდილება. აკეთებს გუნდის სისტემების შემოწმებას უსაფრთხოების სისუსტეებზე შეღწევადობის ტესტების მეშვეობით. | 8 | 7% |
| C# and .Net framework | აქვს C# და .Net framework-ის საბაზისო ცოდნა:   * Language Basics (Syntax, Types, Expressions and Operators, Statements, Namespaces) * Classes, Inheritance, Access Modifiers, Structs, Interfaces, Enums, Generics, Records, Tuples * Collections (Arrays, Lists, Dictionaries, Queues, Stacks and Sets) * LINQ (Queries and Operators) * Delegates (Func, Action) * Events * Lambda Expressions * Exceptions and handling (try…catch) * Nullable value and reference types * Anonymous types * Attributes * Operator overloading * Extensions methods * Pattern matching * Dates and Times * String formatting and parsing * Equality and order comparison * Streams and I/O * Networking (HttpClient) * Serialization (XML, JSON)   ძირითადად დამოუკიდებლად შეუძლია მარტივი და საშუალო სირთულის ამოცანების შესრულება. | აქვს .Net framework-ის შემდეგი ბიბლიოთეკების ძირითადი საკითხების ცოდნა და შეუძლია ამოცანის შესრულება:   * ASP.Net Core: Web API, Controllers, Actions, Middleware, App startup, Filters & Attributes, Routing, Validations, Versioning, Configuration and Options, Logging, Error Handling, Swagger / OpenAPI specification, Localization, Hosting and deployment * Dependency Injection, Service registration methods, Service lifetimes * Concurrency and Asynchrony basics: Threading, Tasks and async/await * SqlClient * Entity framework (Core) basics: Database Context, Code First & Entity modeling, Entity configuration, Querying, Saving, Object and Change tracking, Migrations * FluentValidations * xUnit, Moq, FluentAssertions, Wiremock.Net * Logging (Levels, Scopes, Serilog, Semantic or structured logging) * Object Mapping (AutoMapper, Mapster, Mapperly)   კოდის წერის დროს იცავს ბანკში შეთანხმებულ სტილს, სტანდარტებს და კონვენციებს. | * Asp.net core: Custom formatters, Custom Model Binding and validations, Conventions, Minimal API, Remote Debugging * Health checks * Advanced Entity framework (Core): Lazy Loading, Eager Loading, Explicit Loading, Inheritance (TPT, TPH) * Dapper * Hosted and Background Services * Caching (Memory and Distributed Caching, Redis) * Regular Expressions * Authentication & Authorization (ASP.NET Core Identity) * Disposal patterns * Diagnostics and Metrics (Debugger attaching and breaking, Debugger attributes, StackTrace and StackFrame, Performance Counters, Activity, ActivitySource/ActivityListener, DiagnosticSource/DiagnosticListener, Windows Event Logs) * Diagnostics Tools (dotnet-counters, dotnet-trace, dotnet-dump, dotnet-gcdump, dotnet-monitor, dotnet-stack) * API clients (RestEase, NSwag) * API Resilience - Timeout, Retry, Circuit Breaker (Polly) * Swagger / Open API Specification custom Operation, Schema & Document Filters * Library Multi-Targeting * Preprocessor Directives | * ASP.Net Core Razor views and pages * Feature Flags * WCF (Client and Services) * Real time communication (SignalR, Web sockets, HTTP/2) * Advanced Concurrency and Asynchrony (Syncronization, Exclusive Locking, Locking and Thread safety, Signaling, Channels, Semaphores) * Parallel programming (PLINQ, Concurrent Collections, Blocking Collection, Task Parallelism) * Data protection (DPAPI) and Cryptography (Hashing, Symmetric and Asymmetric Encryption, PKI) * Assemblies and Reflection * Garbage Collection (Generations, Disposal and Finalizers, Memory Leaks, Weak References) * Dynamic Programming (Dynamic Objects, Dynamic Language Runtime) * BenchmarkDotNet * .NET Performance and Memory profiling (JetBrains dotMemory, JetBrains dotTrace) * Advanced dependency Injection, Scrutor * Advanced Networking (TCP, DNS, FTP) * Advanced LINQ, Expression Tree building and modification | * gRPC (Client and Services, Protobuf) * Native and COM Interoperability * Roslyn (Analyzers, Scripting, Runtime compilation) * Code Generators * Span, Memory * GraphQL (HotChocolate) * OData * BDD (SpecFlow) * AsyncAPI | 10 | 9% |
| SQL Server | * Basic Querying * Data types * Data modelling, creating and modifying basic database objects (Schemas, Tables, Views), their relationships and constraints * Inserting, Updating, Deleting records * Aggregations and Grouping | * Normalization and Denormalization * Advanced Querying (Subquery, PIVOT, UNPIVOT, CTE, ...) * Windowing Functions * Transactions * T-SQL basics: Variables, Control-of-flow, Operators, Stored Procedures, User-Defined Functions and Types | * Advanced T-SQL: Cursors, Hints, Error handling and Logging, XML, JSON, Global and local temporary tables * Query Profiling and execution plan analysis * Concurrency, Isolation levels, Locking and Deadlocking * SQL Server Integration Services (SSIS) basics:   Control Flow and Data Flow basics, Variables, Parameters, Expressions, Database and File sources and destinations, Basic transformations, Error handling and Logging | * Index Architecture, Behavior and Analysis * Managing Indexes and Statistics * Bulk operations (SqlBulkCopy) * CLR Procedures and Functions * In-Memory OLTP * Advanced SQL Server Integration Services (SSIS): Script tasks and script components, Web service and other advanced sources and destinations, Loop containers, Transactions, Incremental Loads, SSISDB catalog deployment, views and procedures | * Extended Events * Database Engine Tuning Advisor; * Query Store * Table Partitioning | 8 | 7% |
| Logging, Telemetry and Monitoring tools | N/A | * Graylog usage basics (Searching logs, Saved searches, Search query language) * Dynatrace usage basics (Applications and services, Request metrics, Problems and Hotspots, Distributed Traces and Backtraces, Searching) | * Graylog configuration (Inputs, Streams, Index sets) * Prometheus metrics and Grafana dashboards usage | * Advanced Graylog (Events, Alerts, Outputs, Dashboards) * Elastic APM usage (Applications, Endpoints, Distributed Traces, Search) * Prometheus metrics and Grafana dashboards, alerts setup and configuration * Dynatrace setup and configuration | * Elastic metrics, alerting and actions * Elastic APM setup and configuration (Dashboards, alerts) * OpenTelemetry collector setup and configuration | 6 | 6% |
| ELK | N/A | * Elasticsearch architecture and fundamentals * Elasticsearch API * Creating Index and Templates * Data types and Mappings * Indexing documents * Text Analysis basics * Data search and Query DSL basics * Kibana basics, dev tools, discover * Elasticsearch.Net and NEST usage in .Net projects | * Advanced Text Analysis (Analyzers, Tokenizers, Token and Character filters, Normalizers) * Advanced Data search and Query DSL * Aliases * Kibana dashboards | * Aggregations (Bucket, Metrics, Pipeline) * Advanced index management: reindex, split, clone, merge, shrink * Elasticsearch Scripting * Logstash (pipelines and plugins: input, output, filter, codec) | * Rolling up historical data * Transforming data * Clustering and HA * Data stream * Kibana metrics * Kibana monitoring, alerting and actions | 4 | 4% |
| Docker & K8s | N/A | * Docker fundamentals * Docker Desktop * Docker CLI * Run and manage containers, mounts, volumes, bindings and networks | * Docker Compose * Create Dockerfiles, build, publish and manage images | * K8s fundamentals * Openshift fundamentals * Openshift CLI and Web Console * Create projects and deploy applications in Openshift * Pods, Replication and other Controllers, Deployment configurations, Image streams, Services, Routes, Config Maps, Secrets * CI/CD pipelines using Docker and Openshift | * Openshift templates * Helm charts * Autoscaling with metrics * Service Mesh, Istio | 2 | 2% |
| RabbitMQ | N/A | * Massage broker basics (Queue, Exchange, Binding, Message, Publisher, Consumer) * RabbitMQ architecture and components * RabbitMQ management UI * RabbitMQ .NET client basics | Massage broker in depth:   * Virtual host * Exchange (Direct, Fanout, Topic, Headers) * Queue (Classic) * Binding * Publisher and Consumer * Message and properties * Acknowledgements and Confirms * RabbitMQ .NET client in depth | * Dead Lettering * QOS * Priorities * Message Lifecycle * Quorum Queues | * Advanced Message Routing * Consistent Hash Exchange type * Sharding Plugin * Shovel plugin * Clustering and HA | 4 | 4% |
| Tibco EMS | N/A | * Massage broker basics (Queue, Topic, Message, Producer, Consumer) * Tibco EMS architecture and components * Tibco EMS management GUI (GEMS) * EMS .NET client basics | Massage broker in depth:   * Queue * Topic * Message and properties * Producer and Consumer * Acknowledgements * Bridge * EMS Administration Tool * EMS .NET client in depth | * Dead Lettering and message redelivery * QOS * Message Lifecycle * Message and Destination advanced properties | * Advanced Message Routing * Fault Tolerance * Clustering and HA | 6 | 5% |