SYLLABUS (AZ)

Programlaşdırmaya giriş

Bu plan C# istifadə edərək backend inkişafı üzrə güclü baza qurmaq istəyən tələbələr üçün hazırlanmışdır.
Tələbələr əvvəlcə əsas İT biliklərini öyrənir, sonra məntiqi və analitik düşünmə bacarıqları formalaşdırır, sıfırdan proqramlaşdırmaya başlayır və sonda real proqram təminatında istifadə olunan peşəkar backend mühəndisliyi mövzularına keçirlər.

Bu ixtisaslaşma aşağıdakıları birləşdirir:

- **Nəzəriyyə + praktiki məşğələlər**
- **Təlimçi ilə labaratoriya (kod tapşırıqlarının izahı)**
- **Aylıq qiymətləndirmələr**
- **Yekun backend layihəsi**

Proqramın sonunda tələbələr backend sistemlərinin necə işlədiyini və layihələrin əsaslarını necə qurmağı tam şəkildə mənimsəyəcəklər.

Qiymətləndirilən Praktiki Suallar

Ev Tapşırıqları:

- 30 test əsaslı suallar
- C# kodlaşdırma tapşırıqları (praktiki tətbiq)
- Hər bir tələbənin balı ayrılıqda qeyd olunur və aylıq performans hesabatına əlavə edilir.

Praktiki məşğələ

Tapşırıqların yoxlanması:

- Ev tapşırıqları və suallar detallı şəkildə izah edilir
- Müəllim çətin və anlaşılmaz mövzuları ayrıca izah edir
- Tələbələr mövzunu tam başa düşənə qədər praktikanı davam etdirirlər

MODULE 1 — IT FOUNDATIONS (Kompüter Elmlərinin Əsasları)

Bu modulda tələbələr əməliyyat sistemləri, proqram təminatı vasitələri, internet texnologiyaları və şəbəkələr haqqında fundamental biliklər qazanacaqlar.

Qazanılan bacarıqlar: IT əsasları, sistem istifadəsi, brauzer təhlükəsizliyi, bulud əsasları, informasiya təhlükəsizliyi.

MÖVZULAR:

1. Əməliyyat Sistemləri:

- Əməliyyat sistemi nədir?
- Microsoft Windows
- Apple macOS
- GNU/Linux — əsaslar, terminal giriş, Linux-un nə vaxt istifadə olunması
- Proqram təminatı (brauzerlər, təhlükəsizlik parametrləri)
- Şəbəkə və Cloud-un əsasları (DNS, əsas protokollar)

2. Proqram Tətbiqləri:

- Proqram təminatının istifadə olunma tələbləri
- Brauzerin təhlükəsizliyi və məxfilik ayarları
- Brauzerin təhlükəsiz istifadə yolları (developer üçün)

3. Kompüter Şəbəkələri:

- Əsas şəbəkə protokolları (HTTP, HTTPS, TCP/IP, DNS)
- Müştəri –server arxitekturası və backend tətbiqlərinin ünsiyyəti

MODULE 2 — Backend İnkişafının Əsasları

Bu modulda tələbələr backend inkişafının əsas prinsiplərini və backend mühəndislərinin məsuliyyətlərini öyrənəcəklər. Modulun sonunda tələbələr sadə layihə planı hazırlaya, GitHub-da repository yarada və backend konsepsiyalarını real ssenarilərdə tətbiq edə biləcəklər.

Qazanılan bacarıqlar: Git & GitHub, Microsoft Copilot ilə debugging, backend prinsipləri, internetin işləmə mexanizmi, debugging üsulları.

MÖVZULAR:

1. Backend Dünyasına Giriş:

- Backend Development nədir?
- Backend inkişafının əsas prinsipləri
- Backend mühəndisinin rolları və məsuliyyətləri
- Tələb olunan bacarıqlar, texnologiyalar və alətlər
- Backend Development Workflow
- Layihə planlaşdırmanın əsasları
- Resurs idarəetməsi və sənədləşmə
- Backend planlamasının effektiv olması üçün inteqrasiya (IDE,JIRA,etc).

2. Layihənin Əsasları

- İnteqrasiya olunmuş kodlaşdırma təcrübəsi (komanda üzvləri ilə əməkdaşlıq etmək, kodun testlərini aparmaq, kod standartlarına riayət etmək)
- Debugging-ə giriş
- Debugging texnikaları
- Microsoft Copilot ilə məntiqi səhvlərin təhlili
- Backend üçün Git əsasları
- İnternet necə işləyir?
- DNS əsasları
- Hosting və Deployment əsasları
- Brauzer necə işləyir?
- HTTP vs HTTPS — təhlükəsizlik və məlumat ötürülməsi

MODULE 3 — Alqoritmlər

Bu modul tələbələrə məsələ strukturları, alqoritmlər və massivlər üzrə problem həll etmə bacarıqları qazandıracaq.

Qazanılan bacarıqlar: Məsələ strukturları, çeşidləmə və axtarış texnikaları, alqoritmik yanaşmalar, praktiki tətbiq.

MÖVZULAR:

1. Məntiqi Düşüncə və Problem Həlli:

- Proqramlaşdırmada istifadə olunan məntiq növləri
- Deduktiv anlayışa giriş
- Decomposition texnikaları
- Top-down və Bottom-up yanaşmaları
- Pseudocode giriş

2. Blok-Sxemlər və Əsas Məlumat Tipləri:

- Alqoritmlərə giriş
- Blok-sxemlər (flow-chart) və alqoritm strukturları
- Praktiki blok-sxem analizi (flow-chart)
- Data tipləri və tiplərin çevrilməsi
- Say sistemləri: 2-lik, 4-lük, 8-lik, 16-lıq (Binary → Octal → Hex)
- Dəyişənlər: declaration, initialization & assignment
- Alqoritm dizaynının əsasları

3. İdarəetmə Operatorları, Dövrlər və Funksiyalar

- IF- Else
- Switch
- Proqramlaşdırmada decision-making
- For, while, do-while dövrləri
- Şərtlərlə dövr istifadəsi
- Funksiyalara giriş
- Funksiya sintaksisi, parametrlər, geri dönüş dəyərləri
- Funksiyalarla problem həlli
- Funksiyalarla strukturlaşdırılmış proqram qurmaq

4. Məlumat Strukturları:

- Massivlər və Linked list-lərin xüsusiyyətləri
- Stack və Queue istifadə halları
- Doğru məlumat strukturunun seçilməsi
- Massiv və Linked list implementasiyası

5. Çeşidləmə və Axtarış Alqoritmləri:

- Axtarış və çeşidləməyə giriş (sorting & searching)
- Linear vs Binary Search (addım-addım)
- Bubble Sort (addım-addım)
- Selection Sort (addım-addım)
- Insertion Sort (addım-addım)
- Merge Sort (addım-addım)

MODULE 4 — C# Proqramlaşdırmaya Giriş

Bu modulda tələbələr həm C# proqramlaşdırmasının əsas, həm də irəliləmiş konseptlərini öyrənəcəklər. Tələbələr yaddaş idarəetməsini, asinxron proqramlaşdırmanı, LINQ, OOP prinsiplərini və C#-ın müasir xüsusiyyətlərini (C# 7–13) dərindən öyrənəcəklər.

Qazanılan bacarıqlar: CLR, Garbage Collector və yaddaş növləri, Obyekt-Yönümlü Proqramlaşdırma, Collections və Generics, LINQ və Lambda ifadələri, Delegatlar və Event-lər, Exception Handling, Multithreading, Tasks, Async/Await, Reflection və Streams, C# 7–13 yeni xüsusiyyətlərinin istifadəsi.

MÖVZULAR:

1. C#

- C# vs C proqramlaşdırması
- Əsas C# sintaksisi: statements, comments, namespaces
- Top-level statements və Global Using directives
- Directives vs statements
- Object sinfi
- CLR, CTS, CLS

- Managed vs unmanaged kod
- Assemblies (EXE, DLL), GAC, NuGet paketləri
- Application Domain vs Process vs Thread
- Garbage Collector (GC):
    - Dispose, Finalize/Destructor
    - GC generations
    - Using statement
    - Yaddaşın idarə olunması

2. Məlumat Tipləri və Operatorlar:

- Var vs Object vs Dynamic
- Implicit vs primitive tiplər
- Float vs Double vs Decimal
- Assignment, Arithmetic, Increment / Decrement
- Relational, logical, bitwise operatorları
- Equals() vs ==
- Unary operator

3. Məlumat Tipləri və Operatorlar (davamı):

- Value vs Reference types
- Stack vs Heap
- String vs StringBuilder
- String formatting & interpolation
- Substring & string manipulation ası
- Boxing və Unboxing
- Implicit vs Explicit casting
- Convert vs Parse vs TryParse
- Type checking: is / as
- Nullable value types
- Null-forgiving operator
- Null-coalescing (??)
- ref, out, ref local, ref return, in parametrləri

4. İdarəetmə Operatorları:

- Selection statements
- Iteration statements
- Ternary operator
- Const vs Readonly

5. Metodlar:

- Metod nədir?
- Extension methods
- Expression-bodied methods
- Tuples
- Nested methods
- Method overloading vs Overriding

6. Obyekt-Yönümlü Proqramlaşdırma (OOP):

- OOP nədir?
- Classes, records, structs, enums
- Fields vs Properties (get/set/init)
- Constructors (private/static/chaining)
- Private vs static constructor
- Deconstructors
- Base vs This keywords
- Static classes
- Anonymous classes
- Sealed classes
- Access modifiers
- Polymorphism vs Encapsulation
- Virtual methods vs method hiding

- Abstract classes & methods
- Inheritance vs interface (C# 8 & 11 dəyişikləri daxil)
- Abstract vs Interface müqayisəsi

7. Massivlər:

- Tək və çoxölçülü massivlər
- Jagged arrays
- Object arrays
- Dynamic arrays
- Indexers
- Partial classes

8. Generics və Collections:

- Generics
- Generic classes, methods və interfaces
- Üstünlüklər və məhdudiyyətlər
- Collections
- Generic vs non-generic collections
- Non-generic: ArrayList, Hashtable, SortedList, Stack, Queue
- Generic: List<T>, Dictionary<TKey,TValue>, SortedDictionary<TKey,TValue>, Stack<T>, Queue<T>, HashSet<T>, LinkedList<T>
- DictionaryEntry vs KeyValuePair
- List<T> vs LinkedList<T>
- Collection initializers
- IEnumerable vs ICollection vs IQueryable

9. LINQ və Lambda İfadələri:

- LINQ əsasları
- LINQ nədir?

- Query syntax vs method syntax
- LINQ operatorları: select, selectMany, orderBy, First, Single, Last, All, Any, ThenBy, Skip/SkipWhile, Take/TakeWhile, TakeWhile vs Where, Where vs Let, AsEnumerable vs AsQueryable
- LINQ və verilənlər bazası
- DataContext rolu
- Niyə SELECT FROM-dan sonra gəlir?
- LINQ vs stored procedures
- LINQ providers
- LINQ növləri (Objects, XML, SQL, DataSet, Entities)
- PLINQ (parallel LINQ)
- Üstünlüklər və məhdudiyyətlər
- Lambda ifadələri: nədir, harada və niyə istifadə olunur, əsas qaydaları
- Expression vs statement lambda

10. Səhvlər və İstisnalar:

- Error nədir?
- Növlər: compile-time, runtime, logical
- Exception nədir?
- Exception sinifləri (DivideByZero, SQL, Format və s.)
- throw vs throw ex
- Exception handling: try/catch/finally
- System.Exception vs ApplicationException
- Inner exceptions
- AggregateException

11. Asinxron Proqramlaşdırma:

- Threads: nədir, növləri, process vs thread
- Thread pool, Multithreading
- Thread metodları: Sleep, Join, Abort, IsAlive
- Locks, Monitor, Mutex

- Deadlock & race conditions
- Tasks: Task.Run vs Task.FromResult, Task.WhenAll, Task.Delay, Task.WaitAll, Task.ContinueWith
- Async/Await
- Async vs sync
- IO-bound vs CPU-bound
- Context switching, Interlocked
- Parallel Programming: Parallel.For, Parallel.ForEach, Parallel.Invoke
- Async vs parallel
- Multithreading vs parallel

12. Delegatlar və Event-lər:

- Delegates: nədir, niyə istifadə olunur
- Multicast delegates və invocation
- Callback functions
- Delegate vs İnterface
- Multicast Exception Handling
- BeginInvoke vs EndInvoke
- Built-in delegates
- Generic vs custom delegates
- Method group conversion
- Anonymous methods vs lambdas
- Event-lər: nədir, publisher/subscriber, EventHandler, async event handling

13. Reflection:

- Reflection nədir
- typeof() vs GetType()
- Reflection istifadə qaydaları və riskləri
- Binding types & binding flags
- MethodInfo & PropertyInfo
- Late binding vs early binding

14. Streams və Fayl İdarəetməsi:

- Stream Class
- Stream properties: CanRead, CanWrite, CanSeek, Length, Position
- Stream metodları: Read, Write, Seek, Close, Flush
- Derived Classes: FileStream, StreamReader/StreamWriter, BinaryWriter, BufferedStream
- Async operations: ReadAsync & WriteAsync
- Buffering & chunk reading
- Exception handling in streams
- Resource management, access control & security
- File & Directory: File vs FileInfo, Directory vs DirectoryInfo

SYLLABUS (EN)

Introduction to the Specialization

This specialization is designed for students who want to build a strong foundation in **backend development using C# .**
Students begin with core IT fundamentals, develop logical and analytical skills, learn programming from zero, and then progress into professional backend engineering topics used in real-world software development.

The specialization combines:

- **Theory + Hands-on practice**
- **Guided labs (coding tasks explanation)**
- **Monthly assessments**
- **One final backend project**

By the end, students will confidently understand how backend systems work and how to create projects fundamentals.

Graded Practice Quiz

Homework Tasks:

- 30 knowledge-based questions
- Coding Exercises (Hands-On Implementation)
- Each student's score is recorded and included in their monthly performance review

Guided Lab

Practice Session:

- Homework questions and tasks are reviewed in detail
- Instructor provides individual explanations to clarify difficult or unclear topics
- Students continue practicing until they fully understand the concepts

MODULE 1 — IT FOUNDATIONS (Computer Science Basics)

In this module students will understand the fundamental knowledge of operating systems, software tools, internet technologies, and networking.

Skills Gained: IT fundamentals, system usage, browser security, cloud basics, information security.

TOPICS:

1. Operating Systems:

- What is an OS?
- Microsoft Windows
- Apple macOS
- GNU/Linux — basics, terminal intro, when to use Linux
- Software Applications (browsers, settings, security)
- Networking & Cloud Basics (DNS, basic protocols, intro to cloud)

2. Software Applications:

- Understanding software usage requirements
- Basic browser security & privacy settings
- Safe browsing practices for developers

2. Computer Networks:

- Key networking protocols (HTTP, HTTPS, TCP/IP, DNS)
- Client–server architecture & how backend apps communicate

MODULE 2 — Foundation of Backend Development

In this module, students will learn the fundamental principles of backend development and understand the responsibilities of backend engineers. By the end of the module, students will be able to create a simple project plan, set up a version control repository on GitHub, and apply essential backend concepts in practical scenarios.

Skills Gained: Git & GitHub for version control, Debugging and logic error analysis using Microsoft Copilot, Core backend development principles, Practical debugging techniques, Understanding how the internet and web requests work.

TOPICS:

1. Introduction to Backend Development:

- What is Backend Development?
- Core Principles of Backend Development
- Roles & Responsibilities of a Backend Engineer
- Essential Skills, Technologies & Tools
- Backend Development Workflow
- Project Planning Fundamentals
- Resource Management & Documentation
- Tool Integration for Effective Backend Planning (IDE,JIRA,etc).

2. Project Development Essentials

- Integrated Coding Practice (collaborate with team members, perform code testing, follow coding standards)
- Introduction to Debugging
- Debugging Techniques

- Using Microsoft Copilot for Debugging & Logic Error Analysis
- Git Basics for Backend Development
- How the Internet Works
- DNS Basics
- Hosting & Deployment Basics
- How a Browser Works
- HTTP vs HTTPS (Security & Data Transfer Basics)

MODULE 3 — Algorithms

In this module, students will build strong problem-solving skills through data structures, algorithms, and performance optimization. By the end, students will be able to design efficient solutions, write algorithmic logic, and analyze performance using Big-O notation.

Skills Gained: Fundamental data structures, Sorting and searching techniques, Algorithmic strategies, Hands-on implementation, Performance analysis

TOPICS:

1. Logical Thinking & Problem-Solving Foundations:

- Types of logic used in programming
- Introduction to deductive reasoning
- Decomposition techniques
- Top-down vs bottom-up approaches
- Introduction to pseudocode

2. Flowcharts & Basic Data Types:

- Introduction to algorithms
- Flowcharts and algorithm structures
- Practical flowchart analysis
- Data types & type conversion
- Number systems: base-2, base-4, base-8, base-16 (Binary → Octal → Hexadecimal)
- Variables: declaration, initialization & assignment
- Basics of algorithm design

3. Control Structures, Loops & Functions:

- If-else statements
- Switch statements
- Decision-making in programs
- For, while, do-while loops
- Using loops with conditions
- Introduction to functions & methods
- Method syntax, parameters & return values
- Combining functions for problem-solving
- Building structured programs using functions

4. Control Structures, Loops & Functions:

- Characteristics of arrays & linked lists
- Use cases for stacks & queues
- Choosing the right data structure
- Implementation of arrays & linked lists

5. Sorting & Searching Algorithms:

- Introduction to sorting & searching
- Linear Search vs Binary Search implementation (step-by-step)
- Bubble sort implementation (step-by-step)
- Selection sort implementation (step-by-step)
- Insertion sort implementation (step-by-step)
- Merge sort implementation (step-by-step)

MODULE 4 — Introduction to Programming in C#

In this module, students will explore both the foundational and advanced concepts of C# programming. They will learn how the .NET runtime works, develop strong understanding of memory management, error handling, asynchronous programming, LINQ, OOP principles, and modern C# features (C# 7–13).

Skills Gained: CLR, Garbage Collector & memory types, Object-Oriented Programming, Collections & Generics, LINQ & Lambda Expressions, Delegates & Events, Exception handling, Multithreading, Tasks, Async/Await, Reflection & Streams, Using new C# features (7–13)

TOPICS:

1. Fundamentals of C# and .NET:

- C# vs C programming
- Basic C# syntax: statements, comments, namespaces
- Top-level statements & global using directives
- Directives vs statements
- Object class overview
- CLR, CTS, CLS
- Managed vs unmanaged code
- Assemblies (EXE, DLL), GAC, NuGet packages
- Application Domain vs Process vs Thread
- Inter-process communication (pipes, shared memory, sockets)
- Intra-process communication (locks, mutexes, events, conditions)

Garbage Collector (GC):

- Dispose, Finalize/Destructor
- GC generations
- Using statement
- Memory cleanup strategies

2. Data Types & Operators:

- `var` vs `object` vs `dynamic`
- Implicit vs primitive types
- Float vs double vs decimal
- Assignment, arithmetic, increment/decrement
- Relational, logical, bitwise operators
- `Equals()` vs ==
- Unary operators

3. Data Types & Operators:

- Value vs reference types
- Stack vs heap
- `string` vs `StringBuilder`
- String formatting & interpolation
- Substring & string manipulation
- Boxing & unboxing
- Implicit vs explicit casting
- Convert vs Parse vs TryParse
- Type checking: `is` and `as`
- Nullable value types
- Null-forgiving operator
- Null-coalescing (`??`)
- `ref`, `out`, `ref local`, `ref return`, `in` parameters

4. Control Statements:

- Selection statements
- Iteration statements
- Ternary operator
- `const` vs `readonly`

5. Methods:

- What is a method?
- Extension methods
- Expression-bodied methods
- Tuples
- Nested methods
- Method overloading & overriding

6. Object-Oriented Programming (OOP):

- What is OOP?
- Classes, records, structs, enums
- Fields & properties (`get/set/init`)

- Constructors (private/static/chaining)
- Private vs static constructor
- Deconstructors
- `base` and `this` keywords
- Static classes
- Anonymous classes
- Sealed classes
- Access modifiers
- Polymorphism & encapsulation
- Virtual methods & method hiding
- Abstract classes & methods
- Inheritance vs interface (including changes in C# 8 & 11)
- Abstract vs interface comparison

7. Arrays:

- Single-dimensional & multidimensional arrays
- Jagged arrays
- Object arrays
- Dynamic arrays
- Indexers
- Partial classes

8. Generics & Collections:

Generics

- Generic classes, methods & interfaces
- Advantages & disadvantages

Collections

- Generic vs non-generic collections
- Non-generic: `ArrayList`, `Hashtable`, `SortedList`, `Stack`, `Queue`
- Generic:
    - `List<T>`
    - `Dictionary<TKey, TValue>`
    - `SortedDictionary<TKey, TValue>`

    - `Stack<T>`, `Queue<T>`
    - `HashSet<T>`
    - `LinkedList<T>`
- `DictionaryEntry` vs `KeyValuePair`
- `List<T>` vs `LinkedList<T>`
- Collection initializers
- `IEnumerable` vs `ICollection` vs `IQueryable`

9. LINQ & Lambda Expressions:

LINQ Basics

- What is LINQ?
- Query syntax vs method syntax
- LINQ operators:
    - `select`, `selectMany`, `orderBy`
    - `First`, `Single`, `Last`
    - `All`, `Any`
    - `ThenBy`
    - `Skip/SkipWhile`
    - `Take/TakeWhile`
- TakeWhile vs Where
- Where vs Let
- `AsEnumerable` vs `AsQueryable`

LINQ and Databases

- DataContext role
- Why SELECT comes after FROM in LINQ
- LINQ vs stored procedures
- LINQ providers
- Types of LINQ (Objects, XML, SQL, DataSet, Entities)
- PLINQ (parallel LINQ)
- Advantages & disadvantages

Lambda Expressions

- What is a lambda expression?

- Where and why we use lambda
- Important rules of lambda
- Expression vs statement lambda
- Type inference
- Does lambda have a type?

10. Errors & Exceptions:

- What is an error?
- Types: compile-time, runtime, logical
- What is an exception?
- Exception classes (DivideByZero, SQL, Format, etc.)
- `throw` vs `throw ex`
- Exception handling: try/catch/finally
- System.Exception vs ApplicationException
- Inner exceptions
- AggregateException

11. Asynchronous Programming:

Threads

- What is a thread?
- Thread types
- Thread vs process
- Thread pool
- Multithreading
- Thread methods: Sleep, Join, Abort, IsAlive
- Locks, Monitor, Mutex
- Deadlock & race conditions

Tasks

- Task.Run vs Task.FromResult
- Task.WhenAll
- Task.Delay
- Task.WaitAll

- Task.ContinueWith

Async/Await

- Async vs sync
- IO-bound vs CPU-bound
- Context switching
- Interlocked

Parallel Programming

- Parallel.For
- Parallel.ForEach
- Parallel.Invoke
- Async vs parallel
- Multithreading vs parallel

12. Delegates & Events:

Delegates

- What is a delegate & why it's used
- Multicast delegates
- Delegate invocation
- Callback functions
- Delegate vs interface
- Handling exceptions in multicast delegates
- BeginInvoke vs EndInvoke
- Built-in delegates
- Generic vs custom delegates
- Method group conversion
- Anonymous methods vs lambdas

Events

- What is an event?
- Publisher/subscriber
- Event handling

- EventHandler
- Async event handling

13. Reflection:

- What is reflection
- `typeof()` vs `GetType()`
- When reflection is useful
- Risks of reflection
- Binding types & binding flags
- MethodInfo & PropertyInfo
- Late binding vs early binding

14. Streams & File Handling

Stream Class

- Stream properties: CanRead, CanWrite, CanSeek, Length, Position
- Stream methods: Read, Write, Seek, Close, Flush

Derived Classes

- FileStream
- StreamReader/StreamWriter
- BinaryWriter
- BufferedStream

Async Operations

- ReadAsync & WriteAsync
- Buffering & chunk reading
- Handling exceptions in streams
- Managing resources
- Access control & security

File & Directory

- File vs FileInfo

- Directory vs DirectoryInfo