**🌱 Phase 1: Days 1–30 — C# Fundamentals & Logic Building**

| **Day** | **Program Title** | **Concepts Covered** |
| --- | --- | --- |
| 1 | Hello World | Basic syntax |
| 2 | Swap Two Numbers | Variables, temp storage |
| 3 | Even or Odd | Conditional statement |
| 4 | Check Leap Year | Nested if |
| 5 | Calculator (Add, Sub, Mul, Div) | Switch case |
| 6 | Print Table of a Number | Loops (for/while) |
| 7 | Factorial of a Number | Loop + math |
| 8 | Fibonacci Series | Loop logic |
| 9 | Palindrome Number | Loop + reverse |
| 10 | Armstrong Number | Math + loop |
| 11 | Sum of Digits | Modulo logic |
| 12 | Reverse a String | Char array, loop |
| 13 | Count Vowels/Consonants in a String | String traversal |
| 14 | Find Length Without Using .Length | Loop logic |
| 15 | Star Pattern (Pyramid) | Nested loops |
| 16 | Inverted Pyramid Pattern | Loop logic |
| 17 | Number Pattern | Nested loops |
| 18 | Basic Calculator using Functions | Methods |
| 19 | Prime Number Check | Loops + condition |
| 20 | Prime Numbers in Range | Nested loops |
| 21 | Sum of Array Elements | Arrays + loops |
| 22 | Reverse an Array | Arrays |
| 23 | Find Largest in Array | Loop + comparison |
| 24 | Sort Array (Bubble Sort) | Sorting algorithm |
| 25 | Count Frequency of Characters in a String | Dictionary |
| 26 | Merge Two Arrays | Arrays |
| 27 | Create a Student Class | OOP - Class/Objects |
| 28 | Create a Bank Account Class | OOP - Properties/Methods |
| 29 | Create a Simple Address Book (Console) | Collections |
| 30 | ATM Machine Simulation | Switch + Methods + Loop |

🌿 Phase 2: Days 31–50 — OOP, Collections, File I/O, LINQ

| Day | Program Title | Concepts Covered |
| --- | --- | --- |
| 31 | Create and Use a Constructor | OOP - Constructor |
| 32 | Inheritance Example | OOP - Inheritance |
| 33 | Polymorphism Demo | OOP - Overriding |
| 34 | Abstract Class Demo | Abstraction |
| 35 | Interface Demo | Interface |
| 36 | Use List<T> with Custom Class | Generics + Collections |
| 37 | Stack Implementation | Stack |
| 38 | Queue Implementation | Queue |
| 39 | Dictionary for Word Frequency | Key-value pairs |
| 40 | File Read/Write Text File | File I/O |
| 41 | Expense Tracker (write to .txt) | File handling + logic |
| 42 | Basic Exception Handling | try/catch |
| 43 | Custom Exception Example | Create your own exception class |
| 44 | Student Filter with LINQ | LINQ basics |
| 45 | Group Products by Category (LINQ) | LINQ + GroupBy |
| 46 | Sort a List using LINQ | LINQ + OrderBy |
| 47 | Find Top 3 Highest Scores (LINQ) | LINQ + Take |
| 48 | Employee Filter (Salary > x) | LINQ with where condition |
| 49 | Mini To-Do App (Console with File Save) | OOP + File + List<T> |
| 50 | Quiz App (Questions from File) | File + Objects + Random |

🌷 Phase 3: Days 51–65 — ASP.NET MVC + Razor + Bootstrap

| Day | Project/Feature | Concepts Covered |
| --- | --- | --- |
| 51 | Setup ASP.NET MVC Project | MVC structure |
| 52 | Set Home Page + Layout | Views, \_Layout.cshtml |
| 53 | ViewBag, ViewData, TempData Example | Razor syntax |
| 54 | Basic Contact Form | Forms, POST/GET |
| 55 | Bootstrap Form Styling | Frontend with Razor |
| 56 | Create Student Model + CRUD (in-memory) | Model-View-Controller |
| 57 | Razor Validation (Required, Email, Range) | Data annotations |
| 58 | Create a To-Do List (Add/Delete Items) | TempData + Loop |
| 59 | Simple Login Page (Hardcoded) | Session, Redirect |
| 60 | Page Layout with Navbar/Footer | Razor partial views |
| 61 | Pass Data from Controller to View (List) | ViewModel + foreach loop |
| 62 | Simple Product List View | Table Display, Model List |
| 63 | Dropdown List for Category | HTML helpers |
| 64 | Add Pagination Logic | Page numbers |
| 65 | Simple Notes App (UI Only) | Forms + Layouts |

🌻 Phase 4: Days 66–80 — Entity Framework + SQL + DB CRUD

| Day | Feature/Project | Concepts Covered |
| --- | --- | --- |
| 66 | Setup LocalDB + EF Code First | DB setup |
| 67 | Create DB Model + Scaffold | Code-first + Migration |
| 68 | Add/Edit/Delete Students (EF) | EF CRUD |
| 69 | Show Student List from DB | ViewModel + DB |
| 70 | Create Product Table + Category Relation | One-to-many |
| 71 | Product Filtering (Category) | EF + LINQ |
| 72 | Create Registration/Login System (EF) | Auth basics |
| 73 | Create Role Table + Seed Data | Role-based logic |
| 74 | Show User Profile (Logged In) | Session + EF |
| 75 | CRUD Blog Posts with EF | Real-life EF CRUD |
| 76 | Notes App with EF | Razor + DB |
| 77 | Expense Tracker with DB | Tables + EF |
| 78 | Add Image Upload (Save Image in Folder) | File upload |
| 79 | Razor Page for User Dashboard | Layout + ViewModel |
| 80 | Basic Chart with JS/Chart.js + DB Values | JS + data visualization |

🌼 Phase 5: Days 81–90 — Frontend + jQuery + Interactivity

| Day | Feature/Project | Concepts Covered |
| --- | --- | --- |
| 81 | Add Client-side Validation | jQuery Validation |
| 82 | Dynamic Dropdown Using jQuery | JS + JSON |
| 83 | AJAX Submit Form Without Reload | AJAX + MVC |
| 84 | Inline Editing of a Table Row | JS/jQuery |
| 85 | Add Sorting by Column | JS Table Sort |
| 86 | Modal Form Using Bootstrap | Modal + POST |
| 87 | Dynamic To-Do List with jQuery | JS + HTML |
| 88 | Use jQuery to Toggle Content | jQuery events |
| 89 | Show Alert/Message Dynamically | Bootstrap Alerts |
| 90 | Use JavaScript to Preview Uploaded Image | FileReader + JS |

🌺 Phase 6: Days 91–100 — Web API + Full Project + Hosting

| Day | Project/Feature | Concepts Covered |
| --- | --- | --- |
| 91 | Create ASP.NET Web API Project | API basics |
| 92 | API Endpoint for Products | GET, POST |
| 93 | Call API from MVC Project (using HttpClient) | API consumption |
| 94 | Create API for Notes CRUD | Full REST API |
| 95 | Create a Mini Blog API | API + DB |
| 96 | Call Blog API using jQuery AJAX | JS + API |
| 97 | Deploy App to IIS (Localhost) | Deployment basics |
| 98 | Publish App to Folder | Project packaging |
| 99 | Upload Project to GitHub | Version control |
| 100 | Final Review: Build Resume/Portfolio Page | Showcase project + confidence booster |

A screenshot of a computer

AI-generated content may be incorrect.

**1. Project Layering Idea**

| **Layer** | **Project Name Example** | **Responsibility** |
| --- | --- | --- |
| Models | YourApp.Models | Data structures, entities, POCO classes |
| Data Access | YourApp.Data | Database context, repositories, EF models |
| Services | YourApp.Services | Business logic, service classes, domain rules |
| API / UI | YourApp.WebApi or YourApp.UI | Presentation layer, controllers, views, API endpoints |
| Tests | YourApp.Tests | Unit, integration tests |

**2. Checklist / Mini-Guide Before You Start a Class or Module**

**Before creating a class/module, ask yourself:**

**Step 1: Define the Layer**

* What layer does this belong to?  
  (Model, Data, Service, UI, Helper, Manager, etc.)

**Step 2: Single Responsibility**

* What is this class/module’s *only* responsibility?
* Will this class do one thing or many things?

**Step 3: Dependencies**

* Does this class depend on other classes/services?
* How will those dependencies be provided? (Constructor Injection? Setter Injection?)

**Step 4: Access & Visibility**

* What should be public, private, internal?
* Who needs access to this class?

**Step 5: Interaction**

* How will other parts of the app interact with this?
* What interface or contract will it expose?

**Step 6: Error Handling**

* What errors/exceptions can happen here?
* How will those be handled or propagated?

**Step 7: Testing**

* How will I test this class?
* Can dependencies be mocked or stubbed easily?

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  
  
A computer screen with a black screen

AI-generated content may be incorrect.