.NET Full Stack Development with AI Integration - Syllabus

A complete path from frontend fundamentals to backend application development - HTML, CSS, JavaScript, React, C#, ASP.NET Core, SQL Server and practical AI integration. 19 core modules, 4 AI modules, 1 full stack capstone project. Duration: 3 Months (flexible schedules available), plus 1 Month placement training.

Part A - Frontend Development (6 Modules)

Course objective
Learn to build responsive user interfaces with HTML, CSS, Bootstrap, JavaScript and React, from page structure and styling to interactive single-page applications.

1. HTML5

Basic HTML structure and document setup, Semantic tags and page structure, Forms, tables and lists, Links and embedding content, Multimedia tags: images, audio and video
LAB: Create a multi-section webpage with a registration form. CHECKPOINT: Structure content for a web page

2. CSS3

Selectors: element, class and ID, CSS box model: padding, margin and border, Layout techniques: Flexbox, Positioning: absolute, relative and fixed, Responsive design using media queries, Animations: transform, translate and keyframes
LAB: Style the HTML page and make it responsive. CHECKPOINT: Style and arrange page elements

3. Bootstrap and Responsive Design

Bootstrap selectors and components, Bootstrap grid system, Layout utilities: Flexbox utilities and grid-based layouts, Responsive design: breakpoints and media queries
LAB: Build a responsive navbar, cards and form using Bootstrap. CHECKPOINT: Develop responsive layouts quickly

4. JavaScript Fundamentals

Variables and data types: numbers, strings, booleans, arrays and objects, Operators: arithmetic, logical and comparison, Control flow: if/else, switch and loops, Functions, parameters and return values, Arrays and array methods, Object manipulation, DOM manipulation: accessing and modifying elements, Event handling: click, hover and change
LAB: Add form validation and interactive DOM behaviour. CHECKPOINT: Add client-side logic and interactivity

5. Advanced JavaScript (ES6+)

Asynchronous programming: callbacks, promises and async/await, JavaScript modules: CommonJS and ES modules, Event handling, forms and routing concepts, Higher-order functions
LAB: Fetch data from an API using async/await and display it. CHECKPOINT: Work with modern JavaScript patterns

6. React JS

JSX syntax and React application setup, Components: functional and class-based overview, Props, lists and keys, Lifecycle methods overview, Conditional rendering, Event handling in React, React Hooks: useState, useEffect, useRef, useContext and useReducer, Routing using React Router, API integration and data fetching using Axios
LAB: Build a multi-page React interface with forms, reusable components and API data. CHECKPOINT: Develop interactive single-page applications

Frontend practice applications
BMI Calculator: calculate BMI from user input, To-Do App: create, update and track daily tasks, Registration Form Validation: validate user details before submission, E-commerce UI: product listings and shopping-cart interface, Weather Dashboard: display weather information using an API.

Learning outcomes
Build responsive web pages and interactive React interfaces. Structure and style content with HTML5, CSS3 and Bootstrap. Add client-side logic and interactivity with JavaScript. Work with modern JavaScript

Greens Technologies

patterns and develop interactive single-page applications with React.

Part B - Backend Development (13 Modules)

Course objective
Develop server-side applications using C#, ASP.NET Core MVC and Web API, and store application data using Entity Framework Core and SQL Server.

7. Introduction to .NET and ASP.NET Core

What is .NET? .NET Framework vs .NET Core vs .NET 5+, Advantages of .NET Core and ASP.NET Core architecture, Development environment setup and .NET SDK installation, Visual Studio / VS Code setup, Project templates overview
LAB: Create and run a basic ASP.NET Core project. CHECKPOINT: Explain the .NET platform and project setup

8. C# Basics (if not known)

C# syntax, variables and data types, Classes, objects and methods, Loops, conditionals and operators, OOP: encapsulation, inheritance, polymorphism and abstraction, Exception handling, collections and LINQ
LAB: Write small C# programs using classes, collections and LINQ. CHECKPOINT: Write basic C# code

9. ASP.NET Core Project Structure

Understanding folders and files, Program.cs and builder pattern, Middleware pipeline overview, appsettings.json and configuration, Dependency Injection concept and implementation
LAB: Explore project structure and register a simple service. CHECKPOINT: Explain application configuration and startup

10. MVC Architecture in ASP.NET Core

What is MVC (Model-View-Controller)?, Creating an MVC project, Role of Controller, View and Model, Routing: attribute and conventional routing, Passing data: ViewData, ViewBag, TempData and Model
LAB: Create a basic MVC application with controller, model and view. CHECKPOINT: Organise an application using MVC

11. Razor Views and Razor Syntax

Razor syntax (@ syntax), Layouts and _ViewStart.cshtml, Partial views and sections, Tag Helpers: asp-for and asp-action, View Components
LAB: Build a form page using Razor syntax, layouts and Tag Helpers. CHECKPOINT: Create dynamic and reusable UI pages

12. Working with Models and Data

Creating Models and ViewModels, Model Binding and Validation, Data Annotations, Entity Framework Core and Code First approach, DbContext and migrations, CRUD: Create, Read, Update and Delete
LAB: Build a CRUD screen with model validation. CHECKPOINT: Perform database-backed operations

13. Database Integration (EF Core)

SQL Server connection, LINQ queries, Relationships: One-to-Many and Many-to-Many, Repository Pattern, Unit of Work Pattern
LAB: Retrieve and display related records using EF Core. CHECKPOINT: Work with relational data

14. Controllers and Actions

Action methods and return types, IActionResult and common result types, JsonResult and Redirect, Filters in ASP.NET Core: Action, Result and Exception filters
LAB: Create actions that return views, JSON and redirects. CHECKPOINT: Explain request handling and responses

15. Dependency Injection and Services

What is Dependency Injection?, Registering services: AddTransient, AddScoped and AddSingleton, Creating custom services, Using services in controllers

Green's Technologies

LAB: Move business logic into a service and inject it into a controller. CHECKPOINT: Separate business logic from controllers

16. Middleware and Request Pipeline

What is middleware?, Creating custom middleware, Built-in middleware: Static Files, Routing and Authorization
LAB: Add a simple custom middleware and inspect request flow. CHECKPOINT: Explain the ASP.NET Core request pipeline

17. Authentication and Authorization

Identity in ASP.NET Core, Login, Register and Logout functionality, Role-based and Policy-based authorization, JWT Authentication for APIs: introduction
LAB: Implement a basic login flow and protect a page or endpoint. CHECKPOINT: Explain authentication and access control

18. ASP.NET Core Web API

What is REST API?, Creating a Web API project, Routing and API controllers, Returning JSON responses, Consuming and testing APIs using Postman, Versioning and CORS: introduction
LAB: Build a simple CRUD API and test it using Postman. CHECKPOINT: Create and test REST APIs

19. State Management

Session and Cookies, TempData, Query Strings, Hidden Fields
LAB: Maintain simple values across requests using Session or TempData. CHECKPOINT: Explain basic web state management

Request flow

Browser / React sends request -> Middleware pipeline steps -> Controller business logic -> Response as View or JSON. Secure login flow: Login form (user credentials) -> Identity (verify user) -> Cookie / JWT (issue token) -> Protected API (role checked).

Learning outcomes

Develop MVC applications and REST APIs using ASP.NET Core and C#. Write basic C# code with classes, collections and LINQ. Perform database-backed operations with Entity Framework Core and SQL Server. Separate business logic using services and dependency injection. Implement authentication, access control and state management. Create and test REST APIs.

Part C - AI Integration (4 Modules)

Course objective

A beginner-level practical introduction to adding AI features to .NET applications. Students use C# and existing AI APIs; deep machine-learning concepts are not included. AI is taught as an application-development skill, not as a machine-learning course.

AI-1. Introduction to AI Integration in .NET Applications

How an application sends user input to an AI service and receives a response, Use cases in enterprise applications, internal business tools, customer support systems and web applications
LAB: Identify one suitable AI feature for a sample business application

AI-2. Integrating OpenAI / Gemini APIs using C#

Calling an AI API from C# using HttpClient, API keys, request/response JSON and basic prompt instructions
LAB: Send a prompt from a .NET application and display the response

AI-3. Basic AI Chatbot, Text Generation and Summarization

Creating a simple chatbot endpoint using ASP.NET Core Web API, Generating text and summarising user-provided content for a business use case
LAB: Build a basic support chatbot or enquiry summarizer

Greens Technologies

AI-4. AI with Application Data and Security

Introduction to AI-powered document processing and intelligent search/recommendation features, Basic overview of using Entity Framework Core and SQL Server data with an AI feature, API authentication, secure configuration, error handling and safe API-key storage
LAB: Add one small AI feature to an existing .NET application

Real-world application

An existing customer-support portal built with React, ASP.NET Core and SQL Server can add an AI assistant. The React screen collects a question, the ASP.NET Core backend validates the request and calls an AI provider securely, and the response is displayed to the user. A business tool may also summarise support enquiries for staff. The application continues to manage user access, business rules and stored records.
Do: Keep API keys in secure configuration, Validate every request on the backend, Handle errors, limits and timeouts, Show clear messages to the user.
Avoid: Hard-code keys or push them to GitHub, Send private user data without consent, Trust AI output blindly for decisions, Call the AI provider directly from React.

Learning outcomes

Understand how AI features fit into a .NET application. Call OpenAI / Gemini APIs securely from C#. Build a basic chatbot and summarization feature. Add one practical AI feature to an existing application.

Part D - Full Stack Integration and Project

How frontend, backend and database work together

React frontend (pages, forms, UI) -> HTTP / REST API requests -> ASP.NET Core backend (controllers, services, business logic) -> Entity Framework Core -> SQL Server database (tables, records, relationships).

Full Stack Integration Practice

Create React pages and forms for a selected application, Create ASP.NET Core Web API endpoints for create, read, update and delete operations, Connect the API to SQL Server using Entity Framework Core, Call backend APIs from React using Axios or fetch, display data and handle form submissions, Test API endpoints using Postman and verify records in SQL Server, Add login/access control basics and demonstrate one practical AI API feature.

Learning outcomes

Connect frontend, backend and SQL Server using API calls and Entity Framework Core. Create and demonstrate a full stack project with an optional practical AI feature.

Capstone Project: Full Stack Application

Choose one integrated project. Suggested projects: Student Management Portal - React frontend, ASP.NET Core API, SQL Server CRUD, validation and basic login. Employee Management System - employee records, department relationships, role-based access and API integration. E-commerce Web Application - product listing, cart interface, order records and backend APIs. AI-enabled Customer Support Portal - support enquiry form, stored tickets and a basic AI response/summarization feature.

Program Details - Tools, Practice and Placement

Duration

3 Months, flexible schedules available. Placement Training: 1 Month - resume, mock interviews and project presentation.

Technologies covered

Frontend: HTML5, CSS3, Bootstrap, JavaScript, React JS. Backend: C#, .NET, ASP.NET Core, MVC, Razor Views, Web API. Database: SQL Server, Entity Framework Core, LINQ. API and Testing: REST APIs, Postman, Axios / fetch. AI Introduction: OpenAI API, Gemini API. Development Tools: Visual Studio / VS Code, browser DevTools, Git and GitHub (project workflow).

Job Placement Training - 1 Month

Resume preparation and project explanation practice, Building and updating Naukri profile, Development placement training, Technical interview preparation, mock interviews and MCQs, Group discussion and communication practice.

Student deliverables

Frontend practice pages and React application source code, ASP.NET Core MVC / Web API exercises and SQL Server database work, One integrated Full Stack capstone project, Project walkthrough and demonstration to trainer, GitHub repository / source code and resume project description.

Greens Technologies