Web-Based Behavioral Group Therapy Application

Language used: F#

Team: Code-Blooded

24th November, 2025

Meet our Team Members

- Sudeepta Bal - 801455628
- Sanjyot Sathe - 801426514
- Chetan Kapadia - 801438508
- Rucha Tatawar - 801420899
- Nidhi Shah - 801420572

Table of contents

What is the language paradigm?

- F# is a functional-first language focusing on functions, immutability and predictable transformations.
- Reduces hidden side effects and simplifies complex workflows.
- Supports pattern matching, discriminated unions, function composition and strong type inference.
- Also supports OOP and imperative styles but functional constructs form the core.
- Well-suited for data modeling, validation, and branching logic.

How does it fit into the history of programming languages?

- F# was created by Don Syme, an Australian computer scientist at Microsoft Research.
- Belongs to the ML family and heavily inspired by OCaml.
- Designed to bring functional programming into the .NET ecosystem.
- Combines academic principles (type theory, static typing) with practical engineering tools.
- Evolved to support data processing, cloud computing, financial modeling, and backend APIs.
- Strong interoperability with C# and .NET libraries.

What did you find different about the language from other languages in the same paradigm?

- Practical + industry-friendly
- Deep .NET integration
- Supports functional + OOP + imperative
- Clean, minimal syntax
- Easy to use with real-world frameworks

What features did you find helpful?

- Clean data models (record types)
- Simple validation & branching (pattern matching)
- Readable flow (lambdas + pipelines)
- Fewer bugs (strong type inference)
- Safer code (immutability)
- Easy .NET integration
- Quick JSON building (anonymous records)
- Smooth concurrency (async workflows)

Useful F# Features in Our Project

1. Lightweight Record Types

- Clean and concise model definitions (Event, User, RSVP)
- Reduced boilerplate and improved readability

2. Functional List Processing

- Powerful operations like List.filter and lambda functions
- Clear and expressive data validation logic

3. .NET Interoperability

- Works smoothly with ASP.NET Core, MongoDB Driver, Swagger, JSON libraries
- Ability to use all .NET APIs while writing idiomatic F#

4. Simple Control & Error Handling

- Minimal syntax for if-else, for loops, and try-with blocks
- Easy to implement validation, looping, and failure responses

Why We Chose F# for Our Project?

1. **Clean and Concise Code:** F# reduces boilerplate and keeps our backend logic clear and compact.
2. **Powerful Functional Features:** Pipelines, lambdas, and list operations made data processing simple and expressive.
3. **Full .NET Ecosystem Support:** We could use ASP.NET Core, MongoDB Driver, Swagger, and JSON tools without any issues.
4. **Strong Type Safety:** F#'s type system catches mistakes early, making the API more reliable.
5. **Flexible Programming Style:** We used object-oriented controllers with functional logic, F# supports both seamlessly.
6. **Simple Error Handling:** Try-with blocks made exception handling clean and easy to manage in API endpoints.

Is F# best suited for one particular type of application?

- F# Functional-first language On the .NET platform
- F# + .NET Core
- F# is especially strong for data-heavy and logic-driven applications.
- F# is well-suited for concurrent and parallel applications, as its immutable-by-default design helps avoid threading bugs and makes high-performance server workflows easier to implement.
- It excels in data analytics, financial modeling, scientific computing, and machine learning workflows.
- While it works for general .NET development, F# shines most when projects require high reliability, correctness, and heavy data manipulation.

Time for Demo!!!

Github link:

https://github.com/sbal2911/SPL_Final_project

Thank You for Your Time!

We appreciate your attention and interest in our F# Web-Based Behavioral Group Therapy Application.