

Smart Habit Tracker CLI Project (TypeScript)

Step 1 – Project Setup

Initialize a new Node project, install TypeScript, and configure `tsconfig.json`. Enable strict mode, `noImplicitAny`, and `strictNullChecks`. Ensure your project compiles cleanly before continuing.

Step 2 – Design Your Domain Types

Create an enum for habit status, an interface for `Habit`, and type aliases where appropriate. Include readonly fields, optional properties, and proper Date typing.

Step 3 – Build the `HabitManager` Class

Create a class responsible for managing habits. Implement methods for adding habits, completing habits, retrieving habits, and generating statistics. Use proper return types and strict null handling.

Step 4 – Create Advanced Types

Define a `HabitStats` type. Improve it using utility types like `Record`, `Pick`, and `Omit`. Create a 'habit preview' type that excludes certain properties.

Step 5 – Add File Persistence

Use Node's `fs` module to save and load habits from a JSON file. Handle Date serialization/deserialization properly and implement a type guard for safe parsing.

Step 6 – Build the CLI Layer

Use a CLI library like commander or yargs. Define command types explicitly. Avoid using 'any'. Ensure commands are strongly typed.

Step 7 – Implement a Generic Repository

Create a reusable generic Repository class with constrained types. Refactor HabitManager to use this repository.

Step 8 – Add Streak Calculation

Implement a function that calculates streaks based on completion dates. Handle edge cases and ensure the function is fully typed.

Step 9 – Add Custom Error Handling

Create custom error classes such as HabitNotFoundError. Implement discriminated union result types for safer error handling.
