# 📘 User Documentation – EasySave v1.0

## 🔷 General Context

**EasySave** is a file backup tool developed as part of the **Software Engineering** module at CESI, within the fictitious company **ProSoft**. The goal is to provide a reliable file copy system following professional software development standards.

This deliverable (v1.0) corresponds to a **console-based C# application**, using the **.NET 8.0** framework, supporting **full and differential backups** and generating **JSON tracking files**.

---

## 🔷 Functional Objectives

- Create, configure, and run **backup jobs**
- Two backup types supported:
  - **Full**: copies all files from the source
  - **Differential**: only copies modified files since the last full backup
- Automatically generates:
  - `log.json`: execution log

- `state.json`: real-time backup status

---

## 🔷 Technical Architecture

- **Language**: C#
- **Framework**: .NET 8.0
- **Type**: Console application
- **Project structure**:
  - `Models/`: business entities (BackupJob, FileDetails, etc.)
  - `Services/`: core logic (BackupManager, LogManager, etc.)
  - `Utils/`: helper methods
  - `Program.cs`: application entry point

---

## 🔷 GitHub Repository

The full source code is publicly available:

🔗 **https://github.com/Lewarde/Genie-Logiciel**

To clone and run the project:

```
git clone https://github.com/Lewarde/Genie-Logiciel.git
cd Genie-Logiciel/EasySave
dotnet build
dotnet run
```

---

## 🔷 Installation & Execution

### Prerequisites:

- Windows 10 or higher
- .NET SDK 8.0
- Visual Studio 2022 (or VS Code)

### Run the application:

```
dotnet build
dotnet run
```

---

## 🔷 Application Usage

The application provides a **menu-based command-line interface**:

- **Create a new backup job**:
  - Enter job name, source path, target path, and type (Full or Differential)
- **Run a backup job**:
  - Select and execute any of the existing jobs
- **Track the process**:
  - `log.json`: stores the history of operations (file name, size, duration)
  - `state.json`: stores real-time status of progress (remaining files, size, etc.)

---

## 🔷 Best Practices

- Avoid editing JSON files manually.
- Ensure all source/target directories exist and are accessible.
- Prefer launching backups during low system activity.

---

## 🔷 Known Limitations (v1.0)

- Console-based interface only (no GUI)
- No scheduling system
- Differential backup relies solely on **last modified date**
- No multi-threading or parallel copy yet

---

## 🔷 Future Developments (v2.0)

- Graphical user interface (WPF – **MVVM** pattern)
- Improved performance (parallel execution)
- Advanced filters (file extensions, size, date)
- Cloud or network backup support