# Signal Booster Assignment

Time: ~1 hour

## 📘 Scenario

You’ve inherited a core utility from a developer who believed in “moving fast and breaking things.” The tool reads a physician’s note, extracts relevant information about the patient’s durable medical equipment (DME) needs such as CPAPs or oxygen tanks and sends the structured data to an external API.  
  
Unfortunately, this developer took minimalism to an extreme:  
- All logic is packed into `Main`  
- Variable names are cryptic and inconsistent  
- The code includes misleading comments and unused logic  
- There’s no logging, no error handling, and no unit tests  
  
Now, it’s your responsibility to clean it up. The business needs this feature to be reliable, maintainable, and production-ready and they need it fast.

## 🧪 Your Mission

Refactor the provided code into something that’s understandable, testable, and maintainable. Specifically:  
  
1. \*\*Refactor the logic into well-named, testable methods\*\*  
 - Improve structure and readability  
 - Remove redundant or dead code  
 - Use clear and consistent naming  
  
2. \*\*Introduce logging and basic error handling\*\*  
 - Avoid swallowing/hiding exceptions  
 - Log meaningful steps for observability  
  
3. \*\*Write at least one unit test\*\*  
 - Show how you’d test a meaningful part of the logic  
  
4. \*\*Replace misleading or unclear comments with helpful ones\*\*  
  
5. \*\*Keep it functional\*\*  
 - Your version must still:  
 - Read a physician note from a file  
 - Extract structured data (device type, provider, etc.)

- POST the data to `https://alert-api.com/DrExtract` (Not a real link)  
  
6. Only if you have time   
 - Replace the manual extraction logic with an LLM (e.g., OpenAI or Azure OpenAI)   
 - Accept multiple input formats (e.g., JSON-wrapped notes)  
 - Add configurability for file path or API endpoint  
 - Support more DME device types or qualifiers

## 📄 README Requirements

Please include a short `README.md` file in your submission with the following:  
  
- What IDE or tools you used (e.g., VS Code, Rider, Visual Studio)  
- Whether you used any AI development tools (e.g., GitHub Copilot, Cursor, Cody)  
- Any assumptions, limitations, or future improvements  
- Instructions to run the project (if needed)  
  
✅ We encourage the use of AI tools to help you complete this assignment part of what we're evaluating is how you integrate modern development practices.

✅ If you are not a C# developer, we want you to re-write this into the language of your choice then follow the above.