**Title**  
Lightweight Blockchain Crawler & Listener

**Team Members**  
Daniel Folino,

**Project Abstract**  
In this project, we will create a simple blockchain crawler and listener node. The tool will listen on a blockchain test network (for example, Eth. testnet) and track and log new blocks and transactions in real-time. This app is a feathery introduction to blockchain connectivity & event handling, offering the ability to explore blockchain data, without the need to fuss with a full node.

**Short, High Level Description of the Project**  
It will create a simple node, use standard libraries (e.g., web3. js or Python’s web3). It will subscribe to new blocks and transactions, recording relevant information in a local log or lightweight database. A command-line or web-based interface will provide real-time updates (e.g., a one-liner showing how blockchain events can be followed and logged)..

**Objectives**

* **What are you hoping to accomplish?**
  + Develop a crawler that establishes a connection to a blockchain test network.
  + Continuously listen for new blocks and transactions, capturing relevant data (block numbers, transaction hashes, timestamps, etc.).
  + Store the gathered data locally for potential further analysis or visualization.
  + Provide a straightforward interface to display real-time blockchain activity.
* **How do you define success?**
  + A working crawler node that reliably connects to the blockchain test network and logs incoming blocks and transactions in real time.
  + Comprehensive documentation outlining the setup, implementation, and usage instructions.
  + A demonstrable interface (CLI or simple web UI) that displays real-time blockchain events to the user.

**Timeline**

* **Week 1:**
  + Set up the development environment and review available libraries (e.g., web3.js or Python’s web3).
* **Week 2:**
  + Develop the basic connectivity module to link the crawler with the blockchain network.
  + Implement event listeners to capture new blocks and transactions.
* **Week 3:**
  + Integrate a lightweight storage mechanism (e.g., SQLite database or JSON logging) for the captured data.
  + Develop a simple interface (command-line or web-based) for real-time monitoring of events.
* **Week 4:**
  + Test the system under continuous operation, perform bug fixes, and refine the codebase.
  + Finalize documentation, prepare a project report, and develop a short presentation/demo.

**Deliverables**

* A GitHub repository containing:
  + Source code for the blockchain crawler and listener node.
  + Scripts for setting up and running the project on a local machine.
* A demonstration interface (CLI or basic web UI) displaying real-time blockchain events.
* A technical report/paper outlining the design, implementation, and lessons learned.
* Documentation including setup instructions and a user guide.
* A blog post or presentation slides summarizing the project outcomes.

**Unknowns and Risks**

* **Current Blindspots:**
  + Limited experience with blockchain network event handling and optimal integration with various APIs.
  + Understanding performance implications and potential API rate limits on test networks.
* **Risks of Success:**
  + The blockchain node connection might face intermittent issues or network delays affecting real-time data capture.
  + Inaccurate data logging due to unexpected changes in test network protocols or API deprecations.
  + Integration challenges between the event listener, storage mechanism, and user interface.
* **Additional Resources Needed:**
  + Guidance on handling asynchronous event-driven programming in the chosen programming language.
  + Mentorship or reference materials on best practices for connecting to and monitoring blockchain networks.
  + Access to robust test network endpoints to ensure continuous data flow during testing.

This project is designed to be approachable for an introductory blockchain class while providing practical exposure to blockchain data interaction and real-time event monitoring.