**Project Proposal: AI‑Driven Vulnerability Prioritization & Remediation System  
Course: CY3002 – Vulnerability Assessment & Reverse Engineering  
Team Members: Muhammad Ahmad Mustafa (22i‑1591), Areej Zeb (22i‑1561)  
Date: 2025‑05‑04**

**1. Objectives**

* **Automated Network & OS Scanning  
  Leverage Nmap to discover open ports, services, and versions on a target host at user‑supplied IPs in real time.**
* **AI‑Based Vulnerability Matching & Prioritization  
  Use NLP embeddings (SBERT) and machine learning to match discovered services to relevant CVE entries (2023–2025), classify each by severity (Low/Medium/High/Critical), and rank them.**
* **AI‑Generated Remediation Guidance  
  Fine‑tune a T5 text‑to‑text model to produce clear, actionable remediation steps for each CVE.**
* **Interactive Web Interface  
  Provide a browser–based GUI for users to enter an IP, view scan results with prioritized CVEs, and request remediation instructions.**

**2. Scope & Deliverables**

| **Phase** | **Days** | **Deliverable** | **Description** |
| --- | --- | --- | --- |
| **1. Proposal & Architecture** | **1–2** | **Project Proposal** | **Define objectives, select tools (Nmap, SBERT, T5, FastAPI), draft system architecture.** |
| **2. Data Collection** | **3–5** | **CVE Dataset** | **Download NVD feeds (2023–2025), extract CVE\_ID, description, CVSS, attack vector.** |
| **3. Preprocessing** | **6–8** | **Preprocessing Scripts** | **Clean/tokenize CVE text (NLTK/spaCy), generate SBERT embeddings, assemble CSV.** |
| **4. Model Training** | **9–11** | **Training Code & Models** | **Train RandomForest severity classifier; fine‑tune T5 for remediation; save artifacts.** |
| **5. Backend Integration** | **12–13** | **FastAPI Service** | **Implement /api/scan (Nmap→CVE mapping) and /api/fix (T5 remediation).** |
| **6. Frontend Development** | **14–15** | **Web UI** | **HTML/CSS/JS interface with IP input, scan button, results table, remediation panel.** |
| **7. Testing & Evaluation** | **16** | **Test Report** | **Functional testing (scanme.nmap.org, local hosts), metrics for classification & generation.** |
| **8. Final Report & Demo** | **17–18** | **Report, Slides, Demo** | **Consolidate documentation, prepare slides, record live walkthrough.** |

**3. Technical Approach**

1. **Nmap Scanning**
   * **Use Python’s subprocess to invoke nmap -p1-1000 -sV <IP>, parse output for open TCP ports & service versions.**
2. **CVE Matching**
   * **Precompute SBERT embeddings for 80 K CVE descriptions.**
   * **At scan time, embed each discovered service description and compute cosine similarity to CVE embeddings; return top‑3 matches.**
3. **Severity Classification**
   * **Train a RandomForest on SBERT‑embedded CVE descriptions labeled by CVSS severity buckets; deploy as part of API.**
4. **Remediation Generation**
   * **Fine‑tune T5‑small on (CVE description → remediation steps) pairs; export as model.safetensors.**
5. **API Implementation**
   * **FastAPI app with CORS; endpoints /api/scan and /api/fix; load models at startup; use background Nmap scans.**
6. **Frontend**
   * **Single‑page app: IP input, Scan button (with spinner & %), results table, remediation textarea & button.**
   * **Debug pane for raw JSON inspection.**

**4. Tools & Dependencies**

* **Scanning: Nmap CLI (v7.95+)**
* **NLP & ML: Python 3.10, sentence-transformers, transformers, scikit‑learn, torch, safetensors**
* **Backend: FastAPI, Uvicorn**
* **Frontend: HTML5, CSS3, JavaScript (Fetch API)**
* **Data: NVD JSON feeds, ExploitDB mapping**

**5. Success Criteria**

* **Accurate detection of open services on arbitrary IPv4 addresses.**
* **Correct mapping of services to relevant CVEs (precision@3 ≥ 0.7).**
* **Severity classifier macro‑F1 ≥ 0.5 on held‑out CVE data.**
* **Remediation generation judged “useful” by manual review (≥ 80% of suggestions actionable).**
* **Responsive web UI with scan and remediation workflows completing within 10 seconds for typical targets.**

**6. Risks & Mitigations**

| **Risk** | **Impact** | **Mitigation** |
| --- | --- | --- |
| **Nmap not on PATH (Windows)** | **Scan fails** | **Use subprocess with full binary path; document install step.** |
| **Large embedding time on startup** | **Slow UI availability** | **Pre‑compute embeddings to .npy; load directly.** |
| **Transformer memory limits** | **OOM on GPU** | **Use CPU inference or smaller batch sizes.** |
| **CORS issues** | **Frontend can’t call API** | **Enable FastAPI CORS middleware.** |