**RikkeiSoft Team: Capstone Project Analysis**

**Your Project:**

**STORYTELLING:**

Imagine a company starting a new app — managers, developers, and users all share ideas, but requirements often get messy, incomplete, or conflicting. Group 1’s project is to build a smart AI assistant that takes natural language input (like meeting notes or documents) and automatically generates clear, structured requirement documents. Their system will produce views tailored for different people (analysts, developers, managers), detect contradictions, suggest solutions, and export everything into professional reports. In simple terms, it’s like having an AI secretary that listens to everyone, organizes their needs, and writes a complete, consistent requirement plan — saving time, avoiding errors, and ensuring no voice is left out.

**----**

The team is building an **AI-powered tool** that uses **Large Language Models (LLMs)** to automatically generate software requirements. Normally, collecting requirements from stakeholders is slow, error-prone, and often misses some perspectives. Their system will take natural language input (like meeting notes or text files) and then:

* Extract requirements,
* Generate different views for different personas (business analyst, developer, manager),
* Detect conflicts between requirements,
* Suggest resolutions,
* Produce professional requirement documents (Word, PDF, Markdown),
* Show usage scenarios and user stories.

The aim: **make requirement engineering faster, more complete, and more inclusive.**

**Team organization**

* You have **five members**, each with clear roles:
  + **Frontend (React, UI/UX)** – builds the user dashboard.
  + **AI/ML & Python** – integrates the LLM and does persona-based generation.
  + **Backend & Database** – manages APIs, storage, architecture.
  + **Full-stack & Quality** – connects frontend & backend, ensures quality testing.
  + **Frontend & Documentation** – supports coding and prepares project reports.

You will use **MS Teams, Zalo, GitHub, Trello, and Google Drive** for meetings, communication, task tracking, and documents.

**Key risks**

* **Lack of LLM expertise** → solved by training and mentor support.
* **Time pressure** (they also have internships) → solved by early deadlines and task redistribution.
* **API costs/limits** → efficient use of APIs and using free tiers.
* **Integration challenges** → early proof-of-concept and cross-training.
* **Scope creep** → strict documentation and demos to keep scope fixed.

**Project scope**

* **In scope:** natural language processing, persona-based requirement generation, conflict detection, dashboards, exports.
* **Out of scope:** multi-language, advanced analytics, real-time collaboration, or custom model training.
* **Future consideration:** enterprise integration, industry-specific personas, advanced workflows.

**Architecture**

Chosen a **3-tier system**:

* **Frontend (React)** → interactive dashboard.
* **Backend (Laravel/PHP)** → API handling, authentication, business logic.
* **AI Processing (Python)** → connects with LLMs, handles NLP, persona generation, conflict detection.
* **Database (PostgreSQL)** → stores documents, personas, and user data.

Rejected: monoliths, microservices (too complex for now), serverless (latency issues), and desktop apps (limited accessibility).

**Development plan**

* **Alpha release (Semester 1):**  
  Setup, authentication, basic LLM integration, persona templates, requirement generation, basic export.
* **Beta release (Semester 2):**  
  Conflict resolution, dashboards, usage scenarios, advanced exports, performance optimization, analytics.

**Quality measures**

* Target: ≥85% accurate requirement extraction, 90% persona consistency, 80% conflict detection precision.
* Speed: ≤8s for generation, ≤25s for document export.
* Reliability: 99.5% uptime.
* Testing: unit, integration, end-to-end, UAT.
* Compliance: GDPR, Vietnam Privacy Principles. ~~Australian Privacy Principles.~~

**Research & Skills**

* They researched existing tools (JIRA, DOORS, Azure DevOps).
* Compared OpenAI vs Hugging Face models.
* Learned about prompt engineering and NLP in requirements.
* Skill gaps: **AI/ML integration, cloud deployment, advanced security, performance optimization**.
  + Mitigation: training, free cloud tiers, tutorials, and mentor support.

**Ethical considerations**

* Privacy: protect sensitive stakeholder data with encryption.
* Bias: ensure personas and AI outputs don’t reinforce stereotypes.
* Transparency: explain how the AI generates requirements.
* Compliance: Australian Privacy Principles, GDPR, IEEE/ISO AI standards.

**QUESTIONS/ANSWERS SESSION**

* Can you explain in your own words why requirements engineering is a good candidate for LLM-based automation?
* How do you see this project being useful for real companies beyond the classroom?

**Your Response:**

1. Requirements engineering is ideal for LLM automation because it's fundamentally a language and communication problem. The process involves capturing scattered ideas from multiple stakeholders, understanding context and intent, identifying patterns, and translating informal conversations into structured specifications. LLMs excel at these tasks - they can process unstructured text, extract key information, understand relationships between concepts, and generate coherent documentation. The repetitive nature of structuring requirements while maintaining consistency across different formats makes it perfect for AI assistance, freeing human experts to focus on validation and strategic decision-making rather than manual documentation.
2. Real companies waste significant time and resources on requirements gathering—studies show 30-50% of project failures stem from poor requirements. Our tool addresses practical pain points: reducing the 20-40 hours typically spent documenting requirements for a mid-sized project, minimizing miscommunication between technical and non-technical teams, and catching conflicts early before they become costly development issues. Startups and SMEs particularly benefit since they often lack dedicated business analysts. The tool democratizes requirements engineering, making professional-grade documentation accessible to teams of any size and ensuring stakeholder voices aren't lost in translation.

**Technical Design**

* Why did you choose a **three-tier architecture** (React + Laravel + Python AI service) instead of microservices or serverless?
* What are the main challenges you expect when integrating Python (AI) with Laravel (backend)? How will you solve them?
* How will your system ensure that the requirements generated for different personas (developer, manager, analyst) remain consistent and not contradictory?

**Your Response:**

**Risks & Constraints**

* You mentioned **API costs and limitations**. What specific strategies will you use to monitor and control these costs?
* If the LLM provides wrong or biased outputs, how will you detect and handle those errors before presenting to users?
* How do you plan to avoid **scope creep**, since this type of project can easily expand with many features?

**Your Response:**

**Quality & Testing**

* You set a goal of ≥85% accuracy for requirement extraction. How will you measure this accuracy in practice?
* What testing strategies will you use for the AI part, since LLMs don’t always give predictable outputs?
* How will you conduct user acceptance testing (UAT) with non-technical stakeholders?

**Your Response:**

**Research & Ethics**

* In your research, what did you find lacking in existing tools like JIRA or DOORS that your system aims to improve?
* How will you ensure compliance with **Australian Privacy Principles** and **GDPR** when dealing with sensitive requirement data?
* What steps will you take to check for **AI bias** in persona-based requirement generation?

**Your Response:**

**Future Vision**

* If you had more time and resources, what “future consideration” feature would you implement first (multi-language, enterprise integration, advanced analytics)? Why?
* How do you see this project evolving into a product that companies might actually adopt?

**Your Response:**