SOFTWARE ENGINEERING & DESIGN (NSED742)

SEMESTER PROJECT

Dr Silas Verkijika

TENDER INSIGHT HUB

Project Overview

Tender Insight Hub is a cloud-native SaaS platform designed to assist South African SMEs in navigating public procurement opportunities. It simplifies complex tender documents, enables meaningful analysis, and helps SMEs assess their readiness to apply for tenders. The system is built as a collaborative, AI-assisted application that supports decision-making and teamwork across multiple user tiers.

The system must be developed using **FastAPI (Python)** for the backend. Teams are required to use **both a SQL database** (such as PostgreSQL or MySQL) and **a NoSQL database** (such as MongoDB or Redis) in the solution to ensure coverage of diverse data storage techniques. The application is **team-based SaaS**, and access is managed per seat across different pricing tiers.

The project is intended to simulate the development of a production-grade application that supports real-world use cases, delivers insights through APIs, and implements high-quality software engineering principles.

Core Use Case Features

1. Keyword Search, Filtering & Match Ranking

- Users can enter free-text keywords (e.g., "road construction", "security services") into a search bar.
- The backend will perform a search on the descriptions of tenders retrieved from the OCDS eTenders API.
- Search results must be filterable by **province**, **submission deadline window**, **buyer (organ of state)**, and **budget range**. The user should be the one to apply filters after the initial search results are returned.

- Once filtered, users can view the tenders remaining after their filter criteria are applied.

2. Company Profile Management

- When a team registers on the system, they are required to create a **Company** Profile.
- Profile fields must include industry sector, services provided, certifications (e.g., CIDB, BBBEE), geographic coverage, years of experience, and contact information.
- Profiles can be updated at any time and are used to drive the readiness scoring and filtering.

3. Tender Document Summarization (AI Integration)

- For tenders with attached documents (PDF or ZIP), the system must extract the text.
- Use appropriate document processing libraries to extract and process the text.
- Use an AI model (such as a HuggingFace transformer or LLM) to generate plain-language summary of the document
- Summaries should emphasize the objective, scope, deadline, and eligibility criteria.
- The system should also highlight any other relevant information as determined by the design team.

4. Readiness Scoring and Suitability Check

- After summarization, users can select “Match This Tender”.
- The system compares the summarized tender requirements to the user’s company profile.
- Output must include:
    - Suitability Score (out of 100)
    - A checklist of matched and unmatched criteria (e.g., “Has required CIDB: YES”, “Operates in Province: NO”)
    - A short recommendation (e.g., “Suitable - low competition expected”)
- Scores must be stored and shown alongside each tender in the workspace teams’ workspace.

5. Workspace & Tender Tracking

- Users can save tenders to their workspace for later action.
- Team members should be able to see all saved tenders when they logon to their workspace.
  There should be an option for tenders to be grouped by status: “Pending Status”, “Interested”, “Not Eligible”, “Submitted”, etc. This status is managed by the team members. Team members should also see who changed the status.
- Each workspace entry should display tender title, deadline, AI summary, and match score.
- Results must be re-ranked dynamically from highest to lowest match score for the user to review most relevant tenders first.
- Teams should be able to collaborate within their workspace (e.g., leave internal notes or task assignments).

Public API Exposure (External Use Cases)

These endpoints must be available for third-party systems and civic tech applications. The API should be documented using Swagger (OpenAPI).

GET Endpoints

- **`/api/enriched-releases`**
  - Returns a filtered list of tenders with metadata, AI summary, and suitability score.
- **`/api/analytics/spend-by-buyer`**
  - Returns aggregated government spending data by organizations advertising tenders.

POST Endpoints

- **`/api/summary/extract`**
  - Accepts a PDF or DOCX file and returns a 120-word summary.
  - Use open-source NLP model or LLM for summarization.
- **`/api/readiness/check`**
  - Accepts a JSON object containing tender ID + company profile.
  - Returns suitability score, checklist, and recommendation string.

SaaS Plan Structure

The system must be multi-tenant and operate on a team (organization) basis. Plans are defined by features and seat limits.

Tier	Description
Free	1 user per team- Search up to 3 tenders/week- No AI summary or export features. No access to Matched Tender function.
Basic	-Up to 3 users per team- Unlimited tender searches- AI summaries and readiness checks enabled.
Pro	Unlimited team members- Access to all features in basic plus exporting of reports from the teams workspace.

Note: Features must be restricted by using programmatic checks on team plan and number of users.

Development Guidelines

Students must follow modern software engineering practices, version control etiquette, and maintain high code quality throughout the semester.

- **Back-End Framework**: All teams must use FastAPI for the backend – development of RESTful APIs.
- **Front-End Technology**: You may use any front-end framework or library (e.g., React, Angular, Vue, plain HTML/CSS/JS).
- **SQL DB:** For storing structured data such as users, profiles, tender metadata, plans, workspace
- **NoSQL DB:** For storing AI summary results, match results, user activity logs, or cached analytics

- **Authentication:** Simulated login for multiple teams; JSON Web Token (JWT) or session-based
- **CI/CD:** Use GitHub Actions for linting, testing, and deployment
- **Version Control:** GitHub repository (1 per team), use branches and pull requests
- **AI Integration:** Use open-source models or free plans only; students must describe their AI pipeline
- **Security:** Basic validation, CORS, error handling, and protection of endpoints required
- **AI Coding Assistance**: Use of tools like GitHub Copilot, ChatGPT, GEMINI or Claude Code is **mandatory,** reflecting current industry practice.

👥 Team Composition & Roles

- Teams must consist of **5–6 students**.
- Each student must **serve as Team Lead** for at least one development cycle or milestone. This should be determined by the teams.
- Each student must submit a **Team Lead Report**, which should cover:
    - Key decisions made during their leadership
    - Challenges faced and how they were addressed
    - Reflections on teamwork and leadership