Academic Year 2025-2026 Semester 1

50002870006 - Java Enterprise Application Development

Assignment A

Requirements

Each team of candidates is required to design and implement an IntelliJ IDEA plugin that provides AI coding assistant features. The assessment will be based on both presentation and submission. All teams will be randomly divided into three groups (namely A, B, and C), while each group is subject to different requirements as below:

Group	Requirements	Submission Due	Presentation Date and Time
A	F0	11.00pm, 28 October 2025	1.30-4.15pm, 29 October 2025
B	F0 + F1	11.00pm, 4 November 2025	1.30-4.15pm, 5 November 2025
C	F0 + F2	11.00pm, 11 November 2025	1.30-4.15pm, 12 November 2025

F0: Basic Functionalities (<u>For All Groups</u>)

Design and develop an IntelliJ IDEA plugin that serves as a teaching assistant (TA) of the *Java Enterprise Application Development* course.

- Implement a course-specific Q&A chatbot with retrieval-augmented generation (RAG) and source citation, which acts as the TA. It pre-processes course materials (such as lecture slides), and breaks them down into searchable knowledge chunks and indexes them. Whenever the user asks a coding-related question, the plugin should retrieve the most relevant chunks from the knowledge base. The retrieved chunks are then provided as specific and contextual information to an LLM service along with the original question. This process enables the TA to generate a sophisticated response that combines general Java programming knowledge with specific and verifiable information from the course materials.
    - As a critical part of this pipeline, for any response generated based on the retrieved course material, the TA must clearly cite the source document and page number/scope. If no relevant material is found and the answer has to be generated based on the LLM's general knowledge, the TA should explicitly state such situation (e.g., *"Response is based on general knowledge; no specific course material is referenced"*).
- User Interface: Create a user-friendly interface that is embedded in the IntelliJ IDEA (e.g. sidebar or tool window) for interacting with the TA.
- Context Awareness: The plugin must be aware of the user's programming context when

generating answers. At a minimum, implement a right-click context menu action in the editor to allow users to quickly ask questions related to the selected code segment.

F1: Extended Feature 1 (For Group B Only)

- Following the above requirement, this feature set focuses on automation of common and repetitive tasks during the coding process. Each team must implement three smart actions as follows:
    - Generating Unit Tests from Requirements: Generate JUnit-based unit tests for a selected Java class or method. The plugin must firstly ask the user with a text input field to describe the specific scenario or requirement they want to test in the form of natural language (e.g., *"test the case where the user ID does not exist"* or *"validate the boundary condition for an empty string input"*). The plugin then processes the user requirement, along with the contextual source code, and invokes the LLM service to generate JUnit test classes that specifically target that scenario.
    - Intelligent Git Commit Message Generator: This feature automates the creation of meaningful Git commit messages following good conventions. The plugin retrieves the code changes (similar to *git diff*) for all staged files, and then sends such diff context to the LLM service, instructing it to analyze the semantic intents of the changes (e.g., a bug fix, a new feature implementation, or a refactoring process). Based on the analysis, the plugin generates a well-structured commit message, preferably following conventional commit specifications, and displays them to the user.
    - Creative Action: Each team must additionally design and implement one creative smart action of their own.

F2: Extended Feature 2 (For Group C Only)

In addition to the basic requirements, this feature focuses on creating a professional user experience for AI-driven source code modification. The system should provide a feature that can perform significant code modification tasks according to the user's instructions, such as refactoring a method by applying a design pattern or fixing a bug in the source code. When modifying the source code, the plugin must firstly display the suggested changes using IntelliJ IDEA's built-in diff viewer, which provides a clear and side-by-side comparison and allows the user to review and approve the AI-generated suggestions before applying them.

References

1. IntelliJ IDEA plugin development: https://plugins.jetbrains.com/docs/intellij/developing-plugins.html
2. If you have any question on IntelliJ IDEA API and plugin development, you may wish to ask questions on https://intellij-support.jetbrains.com/hc/en-us/community/topics/200366979-

IntelliJ-IDEA-Open-API-and-Plugin-Development. There will be volunteers to answer your question.

3. LLM API:

   Aliyun bailian: https://bailian.aliyun.com

   DeepSeek: https://www.deepseek.com

   Teams are encouraged to use free LLM services.

4. LangChain4j: https://github.com/langchain4j/langchain4j

5. Apache Tika: https://tika.apache.org

6. Conceptual Guide to RAG: https://aws.amazon.com/what-is/retrieval-augmented-generation

Submission

Each team is required to submit a package consisting of the following materials:

a) The complete set of source code;

b) One project report that covers the following contents in the given order:

   1) Full names and matriculation numbers of all team members, contact number, email address, and other particulars if any;

   2) The complete list of system functionalities and features implemented;

   3) A simple user manual (including UI samples and descriptions);

   4) System architecture and components, general workflow, major data storage and structures for supporting RAG (e.g., how documents are chunked, indexed, and retrieved), key algorithms, major supporting techniques, LLM integration and prompt engineering, design rationales, and efficiency considerations;

   5) Other technical details and/or information, if any.

   Note: The report should be written in English; file format for submission: PDF.

c) The presentation slides, with the first slide presenting the full names and matriculation numbers of all team members.

   Note: The slides should be composed in English; file format for submission: PDF.

Please place the source code (a) and the documents (b and c) into different folders, compress them into one ZIP file, and submit it via *Canvas*.

Presentation

Each team is required to present the project within 20 minutes, followed by a Q&A session. The presentation should be delivered in English. Candidates are required to:

a) Explicitly demonstrate all implemented functionalities and features;

b) Present the architecture and components, general workflow, major data storage and structures for supporting RAG, key algorithms, major supporting techniques, LLM integration and prompt engineering, design rationales, and efficiency considerations, as well as other issues that are worth discussing;

c) Present the mechanisms, methods and skills of collaboration among team members;

d) Discuss important issues involved during the design and implementation, as well as the corresponding solutions.

Grading Criteria

Grading Criterion	Weight
Implementation of functionalities and features	35%
Technical design and considerations	35%
Presentation quality	10%
Source code quality	10%
Documentation quality	10%

THE END