Topic 5 - Code Snippet Manager

1. Introduction

In this assignment, you and your team will develop a Code Snippet Manager, a tool designed to provide a structured and scalable way to organize, retrieve, and manage reusable code snippets.

2. Problem Statement

Developers frequently reuse and share code snippets in their daily work.

- In microservice architectures, certain pieces of code are intentionally repeated to reduce dependencies and allow flexibility for project-specific changes.
- Infrastructure code such as `Dockerfile`, `docker-compose.yml`, `application.yml`, or AOP aspect code often appears across multiple projects.

Developers also need effective ways to share knowledge. For example:

- **Daily commands or scripts**: Commands like `git log --oneline --pretty` provide a clean, graph-like version history. However, remembering or retyping them each time is cumbersome. Constantly asking ChatGPT or searching online wastes valuable time.

- **Study purposes**: When exploring large codebases, developers often draw UML diagrams (Class Diagrams, Sequence Diagrams, etc.) and take notes to aid their understanding.

- **Documentation**: Developers who care about clean code may want to preserve and share elegant solutions they have crafted.

This raises a key question: **How do we know which snippet to reuse?**

Most existing approaches depend on a developer's memory and how well they organize their personal files. Unfortunately, ad-hoc storage methods, such as text files, browser bookmarks, or random repositories, fail once snippets grow in number and variety.

Without a proper system, developers face challenges such as:

- Wasting time searching for snippets scattered across multiple sources.
- Copy-pasting unverified code without metadata, context, or history.
- Difficulty collaborating or maintaining snippet versions.

You are therefore tasked with creating a robust snippet management application that can store, retrieve, and organize snippets efficiently. In addition, the application should support advanced developer needs such as search, categorization, and version tracking.

3. Basic Features (8 Marks)

3.1. Manage Snippets (2 marks)

Alert
To score full marks (2), you must implement all the features listed below.

Introduction
A Code Snippet Manager must allow developers to manage their snippets. This is the core functionality of the application.

Features

1. The application shall allow developers to create, update, and delete snippets with attributes defined in the **Attribute Table**.
2. The application shall save changes automatically and periodically to prevent data loss. The auto-save interval shall be set to 5 minutes.
3. The application shall prompt developers to save changes before exiting.

Attribute	Details	
Title	Description	A human-readable title for the snippet.
	Mandatory	Optional
	Multiplicity	1/ snippet
	Examples	"*QuickSort Implementation*", "*Dockerfile for Java App*"
Language	Description	The programming language of the snippet.
	Mandatory	Optional
	Multiplicity	1/ snippet

	Examples	*Java, Python, C++*
Tags	Description	Keywords or labels to classify snippets.
	Mandatory	Optional
	Multiplicity	0 or more/ snippet
	Examples	*dockerfile, java, sorting*
Description	Description	Short explanation of what the snippet does or when to use it.
	Mandatory	Optional
	Multiplicity	1/ snippet
	Examples	*"This snippet sets up a Spring Boot app in Docker"*
Code Packages	Description	A package or namespace to uniquely identify the snippet in a large project. Helps avoid conflicts with similar file names.
	Mandatory	Optional
	Multiplicity	1/ snippet
	Examples	*java.util, org.example.utils*
Code Body	Description	The actual code snippet. When it is presented to the developers, it has to be well-formatted with the original indentation.
	Mandatory	Yes
	Multiplicity	1/ snippet
	Examples	*java\nSystem.out.println("Hello World");\n*
Created/ Updated	Description	System-generated record of when the snippet was created and last modified.

Timestamp	Mandatory	Yes (auto)
	Multiplicity	1/ snippet
	Examples	*2025-10-03 14:23:10*

Attribute Table

Notes

- Mandatory: Developers must provide data for this field.
- Optional: Developers may choose whether or not to provide data for this field.

3.2. Retrieve and Search (4 marks)

Alert
To score full marks (4), you must implement all the features listed below.

Introduction
Developers usually read and use code more often than they create reusable snippets. Therefore, supporting powerful retrieval logic is one of the most essential functionalities of the Code Snippet Manager. This part is more challenging than the others, which is why it carries higher marks.

Features:

1. The application shall allow developers to search by title, tags, language, description, code body, code packages, created date, and modified date. The details are described in the *Search Criteria Table*.
2. The application shall allow developers to filter search results by tags, languages, and created date range and modified date range.
3. The application shall allow developers to sort search results by title, tags, languages, created date and modified date.
    a. Sorting shall support both **alphabetical** and **chronological** order, in either **ascending** or **descending** direction.
4. The application shall display a maximum of 25 results per search to ensure readability and performance, instead of returning all matches at once.

Search Criteria	Requirement
Title	• Support **partial matches** (e.g., searching "Java" should return all records where the title, tags,
Tags	

Language	language, description, code body, or code packages contain “Java”). ● Support a **case-insensitivity toggle** so developers can enable or disable case-insensitive search.
Description	
Code Body	
Code Packages	
Created Date	Support date range search: ● If only a start date is provided → return all records from that date onward. ● If only an end date is provided → return all records up to that date. ● If no start or end date is specified → default to showing records from the last 1 month. ● If both start and end date is specified → return all records in the interval.
Modified Date	

Search Criteria Table

3.3. Organization (2 marks)

Introduction
Beyond adding and searching, the organization of snippets is crucial. Regardless of the underlying storage mechanism (e.g., database, CSV, JSON, YAML, TOML, XML, or file-based directories), developers should feel that their snippets are well-structured and easy to browse.

Features:

1. The application shall allow developers to use **nested tag categories**.
    a. Example: `Algorithm/Sorting/QuickSort`
    b. When searching for `Algorithm/Sorting`, the system shall also return all sub-tags such as `Algorithm/Sorting/BubbleSort` and `Algorithm/Sorting/QuickSort`.
2. The application shall allow developers to **categorize snippets into collections**.
    a. Each snippet may belong to **one collection**.
    b. Collections serve as higher-level groupings (e.g., "*Docker Configs*", "*Java Utilities*")。
3. The application shall support **backlinks between snippets**, enabling cross-references.
    a. Example: *Snippet B cites Snippet A* → developers can navigate directly from **B back to A**.

4. Extra Features (4 marks)

You may implement any combination of the following extra features to achieve up to 4 marks in this section.

Mark(s)	Name	Criteria
2.5	GUI	To score full marks, you should implement a desktop GUI (JavaFX, Swing, etc.) or a web-based UI (React, Angular, Vue, Thymeleaf, etc.) covering all basic and extra features. The 0.5 mark is given for UI attractiveness.
2	Snippet Version History	To score full mark, the application should • Track changes made for each save. • Allow developers to track the changes for each code snippet. • Allow developers to revert back to the previous version for each snippet. • Allow developers to compare between two versions of code snippet for the same code snippet.
1	Statistical Result	To score full mark, the application should display the following statistics: • Number of snippets added per week/month. • A **heatmap-style view** (like GitHub contributions) showing daily snippet additions. • Number of snippets by **language**. • Number of snippets by **tags**.

0.5	Export/ Import	To score full mark, the application should: ● Allow developers to export all the current code snippets in JSON/XML. ● Allow developers to import all the current code snippets in JSON/XML.
1	Multi-User Accounts	To score full mark, the application should: ● Implement Username/Password Login. ● Encrypt the user password in your chosen storage mechanism (database, csv, etc.) ● Allow users to change their password, and their new password cannot be the same as the any old password. ● Should enforce a strong password policy: at least 8 characters, including at least 1 special character, 1 digit, 1 uppercase, and 1 lowercase letter.
1	Syntax Highlighting	To score full mark, the application should: ● Apply syntax highlighting based on programming language. ● Provide at least 3 themes for users to choose from (e.g., light, dark, solarized).
1	Comments	To score full mark, the application should: ● Allow users to add comments to selected sections of code within a snippet. ● Multiple comments may be attached to different parts of the same snippet.
1/ each	Other features	We value and appreciate any creative ideas. 1 mark will be given for each feature as long as you can justify:

		1. How this feature makes the Code Snippet Manager more useful 2. How the feature is implemented. 0.5 mark will be deducted for each feature if justification is weak or missing.

5. Sample Input and Output

Providing a sample input and output here might unintentionally limit your creativity. My ideas may not always be the best, and you — with fresh perspectives and ambition — may come up with better solutions to the question:

"How can we organize code snippets for reusability and study?"

For more examples of input and output, you may refer to 7 Best Code Snippet Managers for Devs 2024. Perform a competitor analysis if you wish to make your product stand out from the rest.

6. Guideline

Please refer to the 2025 FOP Guideline (Topic 5) for more details. It contains all the information you need for this assignment. You may also refer to the following repository for a reference to help you get started: https://github.com/The-Missing-UMCS/skeleton.

7. Checklist for Lecturers, Demonstrators, and Students

The features listed under both *Basic Features* and *Extra Features* serve as a checklist to verify whether the application functions as intended.

This checklist provides a transparent and objective measurement for all stakeholders (lecturers, demonstrators, and students) and facilitates the marking process.

8. Question

We must acknowledge that there is always a gap between *what the client wants*, *what the client expresses*, *what developers understand*, and *what developers implement*. Therefore, it is completely normal to ask questions when requirements appear vague.

You may also seek technical advice on introductory skills such as whether to use Spring Boot, how to use Git, etc., as these are common challenges for CS freshmen. Asking such questions is totally understandable. If you continue to struggle with these skills after trying your best through ChatGPT or YouTube, feel free to approach me. I will do my best to support you and your team.

9. Contact Me

If you have any inquiries or in need of further clarification for this assignment, do not hesitate to contact me, Ng Zhi Yang, through WhatsApp (+6017-7809398)