INDIVIDUAL ASSIGNMENT (10%) & GROUP PROJECT BRIEFING (20%)

LEARNING OUTCOMES:

- CLO3: Design and develop a small-scale software application with UML class diagram for GUI-based application.
- CLO4: Participate effectively in group work to construct useful computer programs.

INTRODUCTION

This project is divided into two separate assessments:

i. Individual assignment (10%)
ii. Group project (5 members maximum per group)

PROBLEM STATEMENT

In the digital era, user-friendly software applications play a critical role in solving everyday problems and enhancing user experiences. Your task is to design a GUI-based solution using JavaFX to address a real-world problem of your choice. Examples include:

- **SOLAT MANAGEMENT SYSTEM**
    - **A system to help a masjid manage, who will be the muazzin, imam, and khutbah reader.**
- **ZAKAT MANAGEMENT SYSTEM**
    - **A system to help masjid manage the distribution of zakat to asnaf. (asnaf Information, amount received, total amount received**
- **CEMETERY MANAGEMENT SYSTEM**
    - **A system to help a cemetery keep track of grave lot in the cemetery (lot information, family etc)**
- **CERAMAH MANAGEMENT SYSTEM**
    - **A system to manage and keep track of ceramah session in a masjid. (the ustaz, topics, categories etc)**
- **DONATION MANAGEMENT SYSTEM**
    - **A system to help masjid keep track of donations received**
- **HAJJ and UMRAH MANAGEMENT SYSTEM**
    - **a system to help a company keep track of bookings (mutawif, doctor, hajj, package, date, etc)**

You are required to brainstorm innovative ideas, ensuring the application's functionality is manageable within the project's scope. Key features should include at least user authentication (user to create an account and login), dashboard that provides overview of the app functionalities and transaction management (add, edit and delete transactions).

TECHNICAL REQUIREMENTS:

1. **Use of Objects and Classes:**
    - Implement appropriate classes and objects to model your application.
    - For example, a "Library Management System" may include classes like Book, Member, and Transaction.
2. **4 pillars of OOP:**
    - Demonstrate the use of the 4 Pillars
    - For example, in a "Student Record System," you might have a Person base class extended by Student and Instructor.
3. **Core Functionalities:**
    - **Add Task: Allow users to enter task details such as name, description, priority, and due date.**
    - **Edit Task: Enable users to select and edit an existing task.**
    - **Delete Task: Provide functionality for users to delete a selected task.**
    - **View Tasks: Display all tasks in a list view, sorted by priority or due date.**
    - **Filter Tasks: Allow users to filter tasks based on their completion status (e.g., show only completed or pending tasks).**
    - **Mark Task as Complete: Add a feature to mark tasks as complete or incomplete.**
4. **ArrayLists:**
    - Use ArrayLists to manage dynamic collections of objects.
    - For example, maintain a list of books, tasks, or students.
5. **File Storage:**
    - **Use a plain text file (e.g., tasks.txt) to store tasks persistently.**
    - **Each task should be saved in the file with all necessary attributes (e.g., name, description, priority, due date, and completion status).**
6. **Save and Load Data:**
    - **Implement functionality to save tasks to the text file whenever a task is added, edited, or deleted.**
    - **Implement functionality to load tasks from the text file when the application starts, so users can continue where they left off.**
7. **File Format Example:**
    - **Each task could be stored on a separate line in the text file, with fields separated by a delimiter (e.g., comma , or pipe |).**
8. **Exception Handling:**
    - **Implement appropriate exception handling throughout the application to handle potential errors and improve user experience.**
9. **File I/O Exceptions:**
    - **Handle exceptions that may occur when reading from or writing to the text file (e.g., IOException).**

- **Display an appropriate error message if the file cannot be accessed or does not exist.**

10. **Input Validation:**
    - **Use exception handling to manage invalid inputs, such as:**
    - **Empty fields (e.g., when adding or editing a task).**
    - **Invalid date format or empty due date fields.**
    - **Display meaningful messages to guide users in correcting their input.**
11. **Other Common Exceptions:**
    - **Handle potential NumberFormatException if numeric inputs are required.**
    - **Ensure proper error handling for any unexpected issues to prevent the application from crashing.**
12. **JavaFX for GUI Development:**
    - Design a functional and intuitive JavaFX GUI that allows users to easily interact with the application.
    - Use appropriate JavaFX components such as buttons, text fields, combo boxes, date pickers, and list views.
13. **GitHub for Version Control and Collaboration (OPTIONAL - BONUS):**
    - Each group are encourage to use **GitHub** to manage their project codebase.
    - Create a private repository for your group.
    - Follow good version control practices, such as:
        - Frequent commits with meaningful messages.
        - Using branches for new features.
        - Pull requests to merge changes into the main branch.
        - Resolving merge conflicts collaboratively.

PROJECT WORKFLOW USING GITHUB (OPTIONAL):

1. **Set Up Repository:**
    - Team leader creates a private repository on GitHub and invites all group members as collaborators.
    - Include a **README.md** file to document the project goals, team members, and instructions for running the application.
2. **Branching and Merging:**
    - Each team member works on their own branch for their assigned tasks.
    - Use descriptive branch names, e.g., gui-design, file-handling, or data-logic.
3. **Pull Requests:**
    - Before merging a branch into the main branch, submit a pull request.
    - Another team member reviews the code for quality and correctness before approving.
4. **Commit Messages:**
    - Write clear and descriptive commit messages, e.g., Added Book class with basic attributes or Implemented file-saving feature for tasks.
5. **Issue Tracking:**
    - Use GitHub's **Issues** feature to document and track tasks, bugs, and enhancements.
6. **Code - Final Submission:**
    - The final version of the code must be in the main branch, with all changes reviewed and approved.

PROJECT DELIVERABLES:

1. **Functional Application:**
    - A fully functional JavaFX application that meets the project's requirements.
2. **Project Report (template will be given in iTa'leem):**
    - The report must document the following:
        - **Introduction:** Overview of the project and proposed solution.
        - **Requirements Specification:** Description of the application's features and functionalities.
        - **Design:** UML use case, class diagrams, and flowchart. Indicate the member in charge of each class diagram.
        - **Implementation:** Key code snippets and explanations, and screenshots of the app.
        - **Conclusion:** Summary of the project and lessons learned.
3. **Source Code (must be submitted if not using GitHub):**
    - Well-documented source code, including inline comments, must be submitted in a structured format.
4. **GitHub Repository (OPTIONAL):**
    - A private repository on GitHub containing:
        - All project source code.
        - A **README.md** with clear instructions for running the application.
        - Evidence of collaborative Git usage (branches, commits, pull requests).

TEAM STRUCTURE AND ROLES:

Each group will consist of **5 members (maximum)**. Appoint one leader. It is recommended to assign specific roles to ensure efficient teamwork:

- **Project Manager:** Oversees project progress, manages deadlines, and handles GitHub repository administration.
- **GUI Designer:** Focuses on creating the JavaFX interface.
- **Backend Developer:** Implements core logic, including objects and inheritance.
- **File Handler:** Manages file I/O operations.
- **Tester/Documenter:** Tests the application thoroughly and prepares the project report.

PROJECT EVALUATION CRITERIA:

1. **Technical Functionality (35%)**
    - Correct implementation of programming requirements.
2. **GUI Design (20%)**
    - User-friendly and visually appealing interface.
3. **Code Quality (10%)**
    - Readable, maintainable, and well-documented code.
4. **Creativity and Innovation (10%)**
    - Originality and practical application of the idea.
5. **Project Report (15%)**
    - Clarity, completeness, and adherence to guidelines.
6. **Project Presentation (10%)**
    - Clarity, organization of presentation, explanation of key points, and equal participation of all group members.
7. **GitHub Usage (BONUS 10%)**
    - Effective use of GitHub, including commits, branches, pull requests, and collaboration.

INDIVIDUAL ASSIGNMENT (10%)

- Each group member must write at least ONE class required to complete the software. The class must NOT be the application (main) class and must differ from the other group members. If you need to use more than 4 classes, divide the tasks equally among members.
- Each member must also provide the UML class diagram for their class(es). However, only 1 report is required, which contains all the UML diagrams drawn by each member. Inside the report, state which class and UML class diagram belong to which member.
- Each member is required to fill out the rubric for the individual assignment and submit it individually (rubric will be given in a separate document).
- All codes must be commented properly

GROUP PROJECT (20%)

- As a group, compile all individual classes written and together, write the main application class.
- The GUI design and the overall program will be assessed as part of a group project.
- Include the screenshots of your app with an elaborated description as part of the report. Refer to the project report template given in iTa'leem.
- Only one rubric per group submission.
- Bonus marks will be given for those who use GitHub as a repository to manage the source code.

SUBMISSION REQUIREMENTS:

- **Submission Deadline:** Week 13
- **Deliverables:**
    - Project report in PDF format (compulsory)
    - Source code in a zip file (compulsory if not using GitHub)
    - GitHub repository link (must be private and shared with the lecturer) - optional
    - Group Project rubric – one submission per group.
    - Individual Assignment rubric – submitted individually.
- **Submission Method:** submit via link given in iTa'leem
- **Project Presentation**: Tuesday & Thursday during class time (groups will be selected at random) – week 14. All members are required to participate.

By completing this project, you will demonstrate your ability to apply object-oriented principles to solve real-world problems, work effectively as a team, and develop user-centric application.

REMINDER:

Referring to the Internet is allowed. HOWEVER any attempt to cheat/use directly the model or project from the Internet will lead to ZERO mark in this project, especially for the Login function where many sample codes are available online. Likewise, you are prohibited to ask ChatGPT or any other AI tools to provide the total solution for you.