UNIVERSITY OF THE PHILIPPINES VISAYAS
COLLEGE OF ARTS AND SCIENCES
DIVISION OF PHYSICAL SCIENCES AND MATHEMATICS

CMSC 128: SOFTWARE ENGINEERING I
1st Semester AY 2025-2026

Prepared by:
Nikko Gabriel J. Hismaña
Instructor, Faculty of Computer Science

LAB INDIVIDUAL ASSIGNMENT 1: Starting Fullstack (To-Do List)

ACADEMIC INTEGRITY

As a student of the University of the Philippines, I pledge to act ethically and uphold the value of honor and excellence. I understand that suspected misconduct on given assignments or examinations will be reported to the appropriate office and if established, will result in disciplinary action in accordance with University rules, policies and procedures. I may work with others only to the extent allowed by the Instructor.

PREREQUISITES

- *CMSC 128 Knowledge*
- *GitHub Account and Individual Lab Repository*
- *Code editor (VS Code)*

A major skill you need to learn as a Software Engineer is continuously practicing programming languages you're familiar with while also learning new technologies that fit with the projects you're working on. As such, for your individual lab assignments, you'll have to use tools you're familiar with and look for new tools that will help you meet the requirements of the software you're developing --- a To Do List web app that you'll continuously work on and, ultimately, deploy.

SYSTEM RESTRICTIONS

Assume that you're creating this application for a client who has specific system restrictions. In this case the client has the following restrictions:

1. Do not use a full tech stack (e.g. MERN), library (e.g. React), or framework (e.g. Angular).
2. The client wants you to develop the **To Do List Web App** using:
    a. Frontend: **HTML-CSS-JavaScript** (you can use a CSS framework like Tailwind or bootstrap)
    b. Backend: **Programming language + Database of your choice EXCEPT PHP+MySQL***

For smaller projects, easy backend options are Flask + SQLite (Flask is a Python web framweork that connects to SQLite database --- similar to how PHP works with MySQL) or Node.JS (Express) + MongoDB Atlas, but you're free to choose whichever backend as long as it meets the system requirements. You can also use a BaaS (Backend as a Service) such as Firebase.* ***Just make sure to choose a backend that is easy to scale and deploy.*

INSTRUCTIONS

1. **Create a PUBLIC GitHub repository** named **cmsc128-IndivProject_<Lastname>.**
2. If needed, utilize **gitignore** to exclude packages, libraries that could simply be installed.
3. Install the required tools for your backend.
4. For your first individual lab assignment, you're tasked to develop a simple **To Do list web application**. It should support the basic CRUD (Create, Read, Update, Delete) operations.
    a. **Minimum Features** (all are required to get a passing score in *Features* rubrics):
        i. To Do List interface
        ii. Add task
        iii. Set Due Date and Time (*string* user input only)
        iv. Delete Task (with confirmation dialog)
        v. Edit Task
        vi. Mark Task as Done
        vii. Data persists after refreshing the browser
    b. **Expanded Features** (all are required to get a perfect score in *Features* rubrics):
        i. Set task priority {High, Mid, Low} (*you can use color tags*)
        ii. Record the timestamp for when the task was added (for sorting)
        iii. Use date picker for selecting due date
        iv. Use time picker for selecting due time
        v. Add a "Task Deleted" Toast with an Undo option when deleting
        vi. Sort by: Date Added, Due Date, Priority
5. To reiterate, it should also have an appropriate, lightweight **database** so that all the **data** created/updated/deleted **persists** even after refreshing the browser/restarting your backend server.
6. Ensure that you follow HCI design practices.
7. Once you have your final version of this activity, create a **README.md** file explaining which backend you chose, how to run the web app, and example API endpoints (for connecting to your database).
8. With the README.md, create a **final commit** named: "**cmsc128-Indiv-Act1-finalX**"

GRADING AND SUBMISSION DETAILS:

1. Upload your GitHub repository link via LMS assignment submission.
2. Submission date will be based on the date of the final commit and the defense, whichever comes later. (i.e. you submitted 09/22 and defended 09/26, then 09/26 will be marked as your activity submission date).
3. The **Submission and Defense Period*** is between 09/22 – 09/26, but you can also submit and defend early with a +2 bonus.
4. **Cut-off date is 09/30.**

**A Google Sheet where you can enter your preferred defense schedule will be provided*

FUTURE ACTIVITIES

After this activity, your next activities will be (Assign 2) Developing User Account Creation and Control (Register, Login, Logout) pages, (Assign 3) Implementing User Account control to the To Do List (e.g. each user can only access their own To Do List), (Assign 4) Refining and Deployment. Keep this in mind when picking your backend.

RUBRICS FOR GRADING				
Criteria	**Excellent**	**Good**	**Satisfactory**	**Needs Improvement**
Features	All minimum and most/all expanded features implemented. No/minimal issues/bugs. (15-13)	All minimum features implemented. Some expanded features implemented. Minor issues/bugs. (12-10)	All minimum features are implemented. A few issues/bugs. (9)	Only some minimum features implemented. Multiple significant issues and bugs. (8-1)
Design	Good UX/UI design, follows HCI design best practices with no/negligible design issues. (5)	Good UX/UI design with minimal UX/UI design issues. (4)	Good UX/UI design. Some significant design issues. (3)	Several and/or significant UX/UI design issues. (2-1)
Defense Questions	All questions answered correctly. (10)	One or two questions not answered correctly. (9-7)	More than 2 questions not answered. (6)	All/most questions not answered correctly. (5-1)
Demo, Understanding of Source Code	Demo shows clear understanding of the tech stack. Able to explain all parts of the program and how it works. (10)	Some issues presenting the demo. Unable to explain some parts of the program. (9-7)	Major issues in the demo but still shows understanding of the tech stack. Unable to explain several parts of the program. (6)	Demo not performed correctly. Unable to recall any code needed to run tech stack. Unable most parts of the program. (5-1)
TOTAL: 40 *(PASSING SCORE: 24/40)*				