# 🧾 Code Review Checklist – Assignment Management System

Use this checklist to review your assigned classmate’s Assignment Management System project. Be constructive and specific in your comments. Your goal is to help your peer improve their codebase, just as they are doing for you.

## 🧠 1. General Impressions

- What does the app do well?  
- Was the README helpful?  
- Was it easy to understand the project structure?

The app encapsulates logic very efficiently, test coverage is excellent, and the overall code quality and structure is very professional.

The “project structure” section was descriptive and useful for understanding folder structure and responsibilities.

## 🧱 2. Architecture & Structure

- Are classes, methods, and folders logically organized?  
- Are responsibilities well-separated (e.g., services vs. UI)?  
- Are there signs of tight coupling or unclear responsibilities?

Folders are organized logically and their structure is explained well in the README

The app is well architected and maintains clear separations of concerns.

I could not find any signs of tight coupling or unclear responsibilities.

## 🧪 3. Testing

- Is there a working test suite?  
- Are both unit and integration tests present?  
- Are test names descriptive and easy to follow?  
- Is there at least one test you didn’t expect — something thoughtful or insightful?

Test coverage is excellent and the tests are well formatted.

XUnit and integration tests are present.

Test names are descriptive and notation is consistent.

The use of reflection to test private methods in the console ui was really interesting – I had no idea you could do that!

## 📘 4. Documentation

- Does the README include project setup, usage, and test instructions?  
- Is there any developer commentary or reflection?  
- Are there XML comments or helpful inline comments in complex areas?

README instructions are clear.

Developer commentary is insightful, though a little hard to find. I would recommend storing weekly writeups either in the root folder or in a separate folder contained in the root folder.

XML comment coverage is top-notch! Comments are helpful and well written.

## 🔎 5. Code Quality

- Is the code readable and consistent?  
- Are names descriptive?  
- Are there areas with long or repetitive code that could be refactored?  
- Any TODOs or debug statements left in the final version?

The code is readable and consistent throughout the project.

Names are descriptive and format is consistent.

I could not find any areas with repetitive code.

All TODOs and debug statements are removed.

## 💬 6. Feedback Summary

Write 3–5 sentences summarizing your thoughts on the project. What was impressive? Where could they improve? What would you suggest they work on next? Be specific and supportive.

Khai did a wonderful job with his project. His code quality is professional, his tests are well-written and their coverage is robust, and the overall architecture is clean.

One thing that stood out to me was his custom console class. I liked how it takes the responsibility of validating user input off of the console ui, thus making the class more loosely coupled. After designing my own version of the console ui, I admire how much better Khai’s approach to validation is to my own.

The only thing I would worry about is developer commentary. I was very impressed by the amount of screenshots present for each week, but moving commentary into a separate folder would help with navigation.