SENG300: Introduction to Software Engineering

Project: Online Multiplayer Board Game Platform [0mG]

Course: SENG300
Due Date: March 7 (P1), March 21 (P2), April 11 (P3), April 11 (Videos)
Instructor: Steve Sutcliffe <steve dot sutcliffe at ucalgary dot ca>
Version: 1.0.1
Weight: 50%

Objective:

Design, plan, critique, implement and test a medium-large sized system while developing group dynamic skills.

Group Size: 20 - 25

You are free to form your own groups. This project is designed to be completed in groups of 20. However, you are free to make a group of any size up to a maximum of 25. Keep in mind that you are still expected to complete all of the deliverables.

Each week starting on Feb. 24, each team member must dedicate at least 45 minutes of work per day for three different days each week. Team meetings do not count towards these work sessions. Team members who do not contribute sufficiently, communicate effectively, or participate regularly will receive an F for their contribution. For example, Student A doesn't start working on the project with the other members until the day before it is due. Even though this student spends the final day working on the project, they will receive an F for their contribution because they did not contribute regularly.

Students who do not communicate with their team at the start of each sprint and/or make meaningful contributions will risk being kicked out of the group.

All group members are expected to understand and contribute to the project. To receive a grade for this project, each group member must submit a video explaining their contributions to the code base.

Your mark for this project will be influenced by several factors (e.g., worklog, git commit log, etc.).

You should not share, collaborate, or download code or solutions from your colleagues outside of your group. However, as mentioned in the Generative AI section, you may use external resources.

Overview:

For this project, the CEO wants you to build an online multiplayer board game platform with a few sample games to showcase the functionality. This project will be in three iterations. Iteration 1 will be the planning stage, where you must design and plan out the features for your project. Iteration 2, you will review and critique the designs and plans of another group. Iteration 3, you will receive feedback and design changes from another team in the company evaluating your work, and you must re-design and implement those changes. You can (and should) begin building your system before Iteration 2; however, be aware that you will need to make changes based on the feedback you receive.

Letter from the CEO

Greetings to the greatest (and almost only) development team left at Y!

I am thrilled to invite you to build the next big multiplayer game platform, Online Multiplayer board Game platform (OMG). This is our dream of creating something akin to Steam, but focused on multiplayer board games and community interaction. This system will become the cornerstone of our extensive game library, offering users a seamless way to play, connect, and compete with each other in a wide range of games.

Our vision is grand: a platform where players can login, create profiles, join matches, track their game history, and rise through the leaderboard rankings. We believe this project will set the stage for a vibrant multiplayer community where users can discover new games, challenge friends, and engage in competitive play across all genres. Think of it as a central hub for digital board games, with an emphasis on matchmaking, player profiles, and community-building features. We'll eventually add in AI, real-world robots to manage our system, and hover trains for our employees to get to work, but we'll leave that for the new year.

I encourage you to **think creatively**: How would your matchmaking system work under heavy load? How can the GUI remain responsive even when multiple games are running simultaneously? What features will help our users stay engaged and keep coming back?

Remember, you don't need to **worry about the database**—our dedicated database team will handle that aspect. But it is **critical** that you carefully design **how and where the database will connect** within your system. We also don't require you to **host this platform online**, but your code should contain **stubs** where the server communication logic will reside.

I know we are asking for a lot, but I believe in your abilities. This project will be an incredible opportunity to stretch your skills, and I am confident that with your help, we can build something truly special—a multiplayer board game platform that will become the envy of the gaming industry.

We are counting on your team to design and implement the greatest system in history. We want to release OMG by the end of the quarter, so get cracking!

Sincerely,

Ellen Dusk
CEO, Y

Project Breakdown:

A platform where users can play different board games (like connect four, checkers, Go, Settlers of Catan, etc.) with each other online, with a focus on multiplayer gameplay, user accounts, and ranking systems.

You will be a part of a large group (about 20 students), so you may want to consider forming smaller sub-groups within your team to handle various tasks.

REQUIREMENTS:

Multiplayer Interface

- Users should be able to log in, create profiles, and manage their accounts.
- Each profile should display games played, player stats, and win/loss records.
- Players should have visibility into other users' profiles to view their ranks, current status, and recent matches.

Matchmaking System

- Implement skill-based matchmaking where players are paired with appropriate opponents.
- Add a leaderboard where the top-ranked players can see how they stack up against the competition.
- Enable players to search for and join ongoing games or queue up for new matches.

Game Hosting

- While the platform won't be hosted online in your version, you'll need to design and demonstrate how an online interface would be integrated.
- You will provide stubs/drivers for essential components, such as the database interface and online hosting logic. This ensures we understand where the platform will connect with other systems once live.

Games for Testing

Build a small set of simple games for testing and demonstration purposes. Implement 3 of the following:

- Chess: Implement the basic rules and turn-based gameplay.
- Go: Provide the core mechanics for placing stones and checking for captures.
- Tic-Tac-Toe: A lightweight game to validate basic functionality.
- Connect Four: For testing multiplayer interactions in real-time.
- Checkers: Useful for testing both player turns and piece movement logic.

Design Architecture

You will not need to implement the database or hosting backend (our database team will handle that). Instead, focus on designing where these interfaces will reside within your architecture.

Provide stub functions to simulate how your system will connect to external services such as:

- User authentication database
- Game data storage (match history, leaderboard)
- Cloud hosting or server infrastructure

Graphical User Interface (GUI)

Build a user-friendly interface where players can:

- Select and join games from the available library.
- View and challenge other players by searching their profile.
- Communicate during gameplay using a basic in-game chat.

POTENTIAL TEAMS:

Game Logic Team: Responsible for implementing the rules and mechanics of each game.

GUI Team: Builds the interface for users to interact with the game boards, such as move pieces, display game stats, and chat with opponents.

Networking Team: Manages the multiplayer aspects of the game, including handling real-time interactions, server-client communication, and synchronization.

Authentication & Profile Team: Implements user login, registration, and profile management, including tracking game history and rankings.

Leaderboard & Matchmaking Team: Implements a matchmaking system for pairing players and tracking scores, win/loss statistics, and ranking systems

Integration Team: Works on integrating the separate systems into one cohesive unit.

Demonstration Videos

You must create a video recording of your final product and individual demonstration videos that showcase your contributions to the project. If your code showcase does not contribute sufficiently to the final project code base, your grade will be severely impacted/not credited. The final product demonstration will be a group submission, and the individual demonstration video will be an individual submission.

The product demonstration video should only demonstrate the use cases and full functionality to the client. Do not go into the code details. Avoid technical jargon; you don't need to reveal anything "behind the curtain." Instead, this should demonstrate what your product does. This is fantastic to have in your portfolio to show future employers.

The individual demo video is your opportunity to showcase the intricate workings of your project to a technically astute audience. Imagine you are in a job interview with senior developers and technical leads, explaining your role in a complex software project. Your demo should highlight your contributions to the project code, focusing on significant sections of the code you developed, explaining their functionality and how they align with the project's overall design. Additionally, discuss how your code integrates with the project's architecture and any design decisions or challenges you encountered.

This demo isn't just a tour of your project; it's proof of your technical prowess and your team's ability to solve real-world problems. This technical demo will play a part in evaluating your final project.

The videos must meet these requirements:

- Videos must be in the *.mp4 format.
- Videos must work with vanilla VLC.
- Videos must be uploaded to their appropriate dropbox (no need to have them in the repos).
- The product demonstration video should be named seng300W25_group_<N>.mp4
- The individual demonstration video should be named <UCID>_project_group_<N>.mp4

Substitute <N> with your group number (no <>'s), and <UCID> with your UCID number (no <>'s).

Technology

IntelliJ (if code is required), Git, Microsoft Word, Microsoft Excel, PDF, a drawing application capable of saving as *.png.

Submission instructions

You must submit your assignment electronically using D2L and GitLab (csgit.ucalgary.ca).

Please submit your solution using the appropriate Dropbox in D2L.

Add all TA's and your instructor as Developers on your GitLab project. You must use Gitlab hosted at https://csgit.ucalgary.ca/ (GitHub, Bitbucket, etc. are not allowed). You must make frequent commits under your ucalgary.ca email address and the first and last name associated with your ucalgary account. If you do not make any commits to this assignment you will not get credit for any work and receive an F for this assignment. DO NOT

get a teammate to make the commits for you. You will not get any credit if another teammate submits work on your behalf.

In D2L, you can submit multiple times over the top of a previous submission. Only the latest submission is kept. Every time you need to re-submit something, you must re-submit all the files. Do not wait until the last minute to attempt to submit! D2L is notorious for glitches and hangups at the 11th hour.

Should your assignment involve code, all code must be completed in Java and executable with JDK version 22. You must not use non-standard libraries. Failure to submit the required .java files in D2L will be treated as a non-submission for the assignment. TAs will not grade .class files and will not make requests for missing files.

Project Iteration 1 Deliverables

This iteration (P1) requires that you put into practice all that you have learned in designing and planning a project. You will need to construct a project task timeline, use case descriptions/diagrams, structure diagrams (Class Diagrams), and any other diagrams/documents you feel will best help you plan your work.

All of the *Group Deliverables* require you to submit on D2L and upload it to your Git Repository. *Individual Deliverables* will be uploaded only to D2L, and each deliverable will have a separate Dropbox.

Group Deliverables

All P1 deliverables are required to be in the D2L Dropbox. There are two Dropboxes for group submissions, a private and a public Dropbox. The public Dropbox should not contain any group or individual names or ids. Additional submission guidelines are outlined below. How well these submission guidelines are followed will influence your group's grade. Additionally, you will need to create a repository for this iteration so it is ready when your team begins to program.

Team Document: Create a team organization markdown file with the filename `team.md` that lists your team members, UCIDs and Names. Add this to your repository and to your group's Private Dropbox.

Gitlab Repository: Create a Gitlab repository using the University's Gitlab server (csgit.ucalgary.ca). Your repository must be named `**seng300-w25-project**`. Create a text file called `**gitlab_link.txt**` that contains a link to your repository and submit that file to the private Dropbox. Ensure ALL TAs and your Instructor are added as Developers on your GitLab Repository for this project.

Planning document(s): Prepare any planning documents that you might need which lists all of the milestones, task completion dates, etc., for the programming part of your project. You should use these documents throughout your project to keep on track. Do not include personal/group info, and submit these to the public Dropbox. You should plan to begin working on the code for your project as soon as possible (especially before the P2 deadline).

Structure Diagram: Include a `class_diagram.png` or `class_diagram.svg` that documents your program's most important, vital or complex parts. Include all the parts you consider vital, but feel free to abstract what isn't essential. Do not include personal/group info; submit this to the public Dropbox.

Use Case Descriptions/Diagram: Create a `use_case_descriptions.pdf` that includes a diagram and the use cases that cover your most important interactions. Do not include personal/group info; submit this to the public Dropbox.

Other diagrams/documentation: Include any other diagrams or documentation that would help a third party understand your development strategy and be able to critique it. Do not include personal/group info; submit these to the public Dropbox.

INDIVIDUAL DELIVERABLES

Below, [UCID] should be replaced with your UCID number without the square brackets. All files should be uploaded to the D2L Dropbox unless otherwise stated. There will only be one Dropbox folder for all individual deliverables.

You can modify the existing *.xlsx file templates using Microsoft Excel, which you can access as a University of Calgary student, either through the Office 365 suite or on the lab computers.

Worklog: Deliver a `[**UCID]_worklog_p1.xlsx**` that records the hourly work you did on the project. You must follow a specific format outlined at the end of this document. Failure to name the document correctly and/or failure to follow the format specifications will result in an F for this section of the project.

Peer Evaluations: Deliver a `**[UCID]_peer_evaluations_p1.xlsx**` that evaluates the members of your team. You must follow a specific format outlined at the end of this document. Failure to name the document correctly and/or follow the format specifications will result in an incomplete/F for this section.

These documents heavily influence your grade for this project iteration, including the peer evaluation you receive from other team members.

Project Iteration 2 Deliverables

For iteration 2 (P2), you will receive the P1 Group deliverables of two teams, and you need to review and critique their submissions, suggest changes, and give general feedback. You will need to provide a grade for each group and justification for the grade you gave. You will learn about critiquing submissions in your tutorials.

GROUP DELIVERABLES

Letter to the Reviewed Team: You must submit a letter to D2L Dropbox directed to the reviewed teams. It should summarize your findings, offer feedback, and include the following:

- **General critique of the design**: Highlight strengths and weaknesses.
- **Diagram corrections/changes**: Annotate or edit the original diagrams if errors are found (e.g., UML diagrams, flowcharts).
- **Feature requests**: Identify **two changes or improvements** to the functionality the original team will need to implement.
- **Constructive feedback**: Suggestions for improving readability, consistency, or design patterns.

Planning Document Review: A detailed critique of their project planning and timeline, including:

- Identification of any **timing issues**, missed milestones, or unrealistic deadlines.
- Highlighting **risks** that could impact completion (e.g., dependencies between tasks, resource allocation issues).
- Suggestions for **improving the timeline** or **reordering tasks** to mitigate risks.

Annotated Documentation with Feedback: **Annotated versions** of the design documents with **inline feedback** (comments, suggested edits), which could include corrections to diagrams, suggestions to improve structure or flow, highlighting missing components/elements, and/or sections that need more detail.

Feature Proposal Document: Descriptions of two requested feature changes that need to include a high-level description of the features, expected impacts on the systems and suggestions for implementation or design.

Grade Letter Proposal: A letter grade that you would give the team that you reviewed. This will form the preliminary grade of the P1, which will be passed on to the TAs for their review. The letter grade must be justified, commenting on the design & documentation, planning & timeline, and originality & functionality. Provide this in the separate D2L Dropbox.

Grade Adjustment Table: An adjustment table for all your team members based on the contract you created at the start of P2. Provide this in the separate D2L box.

E.g., Grade Letter Proposal

Proposed Grade: B

Design & Documentation:

Your system design is well-structured, and the diagrams effectively communicate the core architecture. However, we noticed a few areas where the flow between components was unclear. For example, the sequence diagram for the login feature could benefit from more detail on edge cases (e.g., incorrect passwords).

Planning & Timeline:

The timeline is mostly feasible, but some dependencies seem unrealistic, such as completing both game logic and GUI in parallel without accounting for integration time. We recommend adjusting this and building in more time for testing.

Etc.

Accountability System Overview for P2

Each team will create and sign an **Accountability Contract** outlining all members' expectations, responsibilities, and consequences for this iteration. This contract will guide how individual performance is assessed and determine any **grade adjustments**. Teams must also submit a **final list of individual adjustments** at the end of P2 to the group Dropbox.

Deliverables

1. **Accountability Contract** (signed by all team members, submitted at the beginning of P2 to the appropriate Dropbox)
2. **Individual Grade Adjustment Sheet** (submitted at the end of P2 to the group Dropbox)

Deliverable 1: Accountability Contract

Instructions for Teams

1. **Contract Content**:
    - **Roles & Responsibilities**:
        - Each member should agree on their assigned role(s) and the scope of their contributions.
        - Define how the team will ensure **equal distribution of work**.
    - **Participation Expectations**:
        - Establish **clear guidelines** for attendance at meetings, communication, and deadlines.

        - Specify how members should notify the team if they encounter **unforeseen issues** (e.g., illness, family emergencies).
    - **Handling Unfulfilled Duties**:
        - Outline a procedure for **reassigning tasks** if someone can't fulfill their duties due to unforeseen circumstances.
        - Include a timeline for when and how tasks will be redistributed among the remaining team members.
    - **Consequences for Lack of Contribution**:
        - Specify **reasonable consequences** if a team member fails to contribute adequately (e.g., failure to meet deadlines, absence from meetings, lack of effort).
        - Suggested consequences could include:
            - **Grade adjustment**: The underperforming member receives a **reduced final grade** based on the group's consensus.
            - **Additional Duties for P3**: The underperforming member may need to take on **double work** in P3, potentially assigned alongside a more reliable teammate.
            - **Assignment of an 'F' Grade**: If a member fails to meet expectations completely, the group may assign an **F** as a final grade (see Adjustment Sheet below).
    - **Dispute Resolution**:
        - Define how conflicts will be managed. For example:
            - Regular team check-ins or vote-based decision-making.
            - Escalating disputes to the TA if needed.

2. **Submission**:
    - The **signed Accountability Contract** must be submitted to the group's Dropbox **before the start of work** on P2.
    - All members must agree to the terms and sign the contract (either digitally or physically).

Deliverable 2: Individual Grade Adjustment Sheet

At the end of P2, the group will submit an **Individual Adjustment Sheet** listing any **adjustments to members' grades** based on the team's evaluation of participation and performance.

Adjustment Sheet Format

Teams must submit the **Name and Adjustment Score** for every member in the group, including those with no adjustments. Use the following rules to fill out the sheet:

1. **Adjustment Range**:
    - **0**: No adjustment (member receives the team's base grade).

- **-1 to -9**: One step reduction per point (e.g., -1 reduces the grade from B+ to B, -2 from B to B-, and so on).
- **-9**: Automatically assigns an **F** grade.

2. **Format Example**:

Name	Adjustment	Reason
Alice Johnson	0	Consistently met deadlines and team expectations.
Bob Smith	-1	Missed two meetings without notice.
Charlie Davis	-9	Did not contribute to any part of the project.
Diana Roberts	-3	Missed assigned tasks & required others to cover.

3. **Submission**:
   - Submit the **final grade adjustment sheet** to the group Dropbox by the P2 deadline.
   - The group must ensure that the **adjustments align with the contract terms**.

Accountability System Process Summary

1. **Before P2 Starts**:
   - Teams draft and sign their **Accountability Contract**.
   - Submit the contract to the group Dropbox.
2. **During P2**:
   - Teams work collaboratively, following the agreed-upon roles and guidelines.
   - Regular check-ins help ensure **members meet their obligations**.
3. **End of P2**:
   - Teams evaluate each member's performance and complete the **Individual Adjustment Sheet**.
   - Submit the **Adjustment Sheet to the group Dropbox** for final grading.

Evaluation Example: How the Adjustments Affect Grades

Assume the team's base grade for P2 is **B+**.

- **Alice (0 adjustment)**: Receives **B+**.
- **Bob (-1 adjustment)**: Receives **B**.
- **Charlie (-9 adjustment)**: Receives **F**.
- **Diana (-3 adjustment)**: Receives **C+**.

P2 Recommended Division of Labor (for 20-Person Team)

Design Review Sub-Team:

- 6-8 members
- Focuses on reviewing diagrams, structure, and core system design.

Planning Analysis Sub-Team:

- 4-6 members
- Review timelines and task dependencies and identify risks.

Feature Proposal Sub-Team:

- 6-8 members
- Draft two feature requests.

Editorial & Documentation Team:

- 2-4 members
- Compiles the letter, report, and annotated feedback into a clear, professional submission

Project Iteration 3 Deliverables

For this iteration we are expecting a fully functional prototype of the platform, with all essential components implemented. This includes documentation and working versions of the games to demonstrate the platform's capabilities. Things that are beyond the scope of this course (databases, online hosting, etc.) are not meant to be implemented; however, you should have the required stubs in place (see the code base given in assignments 2 & 3).

GROUP DELIVERABLES

Team Document: Submit your team organization document, with the filename `**team.md**`, that lists your team members, UCIDs and Names. Submitted to the D2L Dropbox and in your Repository.

Gitlab Repository: Continue to develop your Git repo. Every team member must contribute something meaningful to the functionality and performance of the game. Failure to do so will result in an incomplete/F for that team member. Submit the link to your Gitlab project in a file called `**gitlab_link.txt**`. **Make sure ALL TAs and your Instructor are added as Developers on your GitLab Repository for this project.**

README File: Include a `**README.md**` file in your Git repository and in the D2L Dropbox, with instructions on how to run the program, a brief description of the game's objective, and an overview of the project's structure.

Git Log File: include a `**git_log.csv**` that documents your team's contributions. See the appropriate section in D2L for instructions.

Changes Made: Include a summary of changes made in your Git repository and D2L dropbox that reviews the changes suggested by the other teams and how you accommodated those changes. Include any before and after sections of the diagrams to make spotting those changes easier.

Video Demo Submission: Submit `group_demo.mp4` video file to D2L. Zoom will allow you to record in that format, and it most likely will be a reasonable size with good quality. If, for some reason, D2L prohibits the upload of the file because it is too large, upload your file to Yuja, and place a file with the link to it called `yuja_video_link.txt` in D2L.

Test Cases: Include the `test_suite.html` that encompasses all unit tests and demonstrates complete coverage of your application.

Information about Yuja here: https://elearn.ucalgary.ca/category/yuja/getting-started-yuja/

INDIVIDUAL DELIVERABLES

[UCID] should be replaced with your own UCID number without the square brackets, e.g., `12345678_peer_evaluations.xlsx`.

Worklog: Deliver a `**[UCID]_worklog_p3.xlsx**` that records the hourly work you did on the project. This file will be submitted only to the appropriate Dropbox, and you are required to follow a specific format. Failure to name the document correctly and/or failure to follow the format specifications will result in an F for this section of the project. The format specifications are available at the end of this document.

Peer Evaluations: Deliver a `**[UCID]_peer_evaluations_p3.xlsx**` that evaluates your team members. This file will only be submitted to the appropriate Dropbox and requires you to follow a specific format. Failure to name the document correctly and/or failure to follow the format specifications will result in an incomplete/F for this section. The format specifications are available at the end of this document.

Individual Video Demo Submission: Submit `[UCID]_demo.mp4` video file to D2L. Zoom will allow you to record in that format, and it most likely will be a reasonable size with good quality. If, for some reason, D2L prohibits the upload of the file because it is too large, upload your file to Yuja, and place a file with the link to it called `yuja_video_link.txt` in D2L.

Reflection: Deliver a `**[UCID]_reflection_p3.pdf**` document that comments on the following using the specified headings:

1. Project Initiation Steps

   List the steps you will take when receiving a new project in a future course. Explain how you plan to approach the project from the initial stages, considering the lessons learned from this course.

2. Group Accountability

   What did you learn from this course about maintaining accountability within a group? Describe strategies or practices that helped you manage group dynamics and ensure collaboration.

3. Design Challenges

   What was the most difficult part of designing a project? Reflect on any specific challenges related to design that you encountered and how you overcame them.

4. Time Management Strategies

   What strategies will you use to ensure your project is completed on time? Discuss approaches to time management and deadlines, including techniques that worked well during this course.

5. Team Formation

   What strategies will you use to form a new team in the future? Consider how you would select and manage team members based on your experience in this course.

6. Design Documentation

What design documents will you use to plan projects in the future? Reflect on which design documentation techniques were most helpful in this course and how you will apply them going forward.

Generative AI Use Evaluation: Complete the End of Term AI Disclosure.

These submissions heavily influence your *individual* grade for the project through both your ability to follow the instructions and the evaluation you receive from your peers.

Group Demonstration Video

Students are required to create a group video demonstration of their system. The video should be as close to 12 minutes long as possible without going over (we will stop the video at 12 minutes). It should cover the key points listed below. The demonstration should be clear, concise, and well-organized, showcasing the functionality and design of the program. It must be in the *.mp4 format that will run with vanilla VLC.

1. Briefly introduce your team (~1 minute).

2. Demonstrate how to set up your program and explain how to run your system (~1 minutes).

3. Demonstrate all of the features of the multiplayer interface (~2 minutes).

4. Demonstrate all of the features of the matchmaking system (~2 minutes).

5. Discuss and demonstrate how the online interface and database will be integrated (~1 minute).

6. Briefly showcase the games for testing (~1 minute).

7. Changes that were made by the reviewing teams and how they were implemented and impacted the system (~2 minutes).

8. Explain and demonstrate the full execution of the test suite (~2 minutes).

Individual Demonstration Video

You are required to make an individual demo video where you go into the details of the code you contributed and how that relates to the system design. This video demo should be no more than 10 minutes in length. There is no minimum length required, but your contributions should be covered clearly and concisely. It must be in the *.mp4 format that will run in vanilla VLC.

1. Introduce yourself (~10 seconds)

2. Code Explanation, Design and Testing (~4 minutes)

- Provide a brief overview of your contributions to the project.
- Highlight any challenges you faced and how you overcame them.
- Explain how your code fits within the overall design of the project and its functionality.
- Highlight the test cases you created, explaining your testing strategy and how it verifies the correctness and quality of your code.

3. Design Reflection and Adjustments (~2 minutes)

- Discuss any divergence from your original planning or design documentation.
- What caused these deviations? Were they due to technical constraints, better approaches, or team decisions?

3. Future Improvement (~3 minutes)

- Highlight any improvements you would make to the code in your system.
- Reflect on what you would do differently based on what you learned during development.

General Notes on Video Submissions

Clarity and Audio Quality: Ensure your audio is clear and your explanation is easy to follow.

Screen Recording: Use Zoom or any other screen recording software to capture your screen and narration.

Conciseness: Stay within the time limit of 10 minutes to keep the presentation concise and focused.

This video will be an essential evaluation component, showcasing technical and presentation skills and will heavily influence your final project mark.

Plagiarism

Discussing the assignment requirements with others is reasonable and an excellent way to learn. However, the work you hand in must ultimately be your work. This is essential for you to benefit from the learning experience and for the instructors and TAs to grade you fairly. Handing in work that is not your original work but is represented as such is plagiarism and academic misconduct. Penalties for academic misconduct are outlined in the university calendar.

Here are some tips to avoid plagiarism in your programming assignments.

1. Cite all sources of code you hand in that are not your original work. You can put the citations into comments in your program. For example, if you find and use code found on a website, include a comment that says, for example:

   ```
   # The following code is from
   https://www.quackit.com/python/tutorial/python_hello_world.cfm.
   ```

   Use the complete URL so that the marker can check the source.

2. A tool like chat-GPT can be used to improve small code blocks. For example, five lines of code. If you get help from code assistance like Chat-GPT, you should comment above the block of code you requested assistance on debugging or improving and cite the tool used to get that suggestion. Using a tool like chat-GPT to write the majority of your assignment requirements will be treated as plagiarism if found without citation, and with citation, it will be treated as 0 for the component the student did not complete. Code improvement of short length will get credit if commented/cited properly.
3. Citing sources avoids accusations of plagiarism and penalties for academic misconduct. **However, you may still get a low grade if you submit code not primarily developed by yourself. Cited material should never be used to complete core assignment specifications. Before submitting, you can and should verify any code you are concerned about with your instructor/TA.**
4. Discuss and share ideas with other programmers as much as you like, but make sure that when you write your code, it is your own. A good rule of thumb is to wait 20 minutes after talking with somebody before writing your code. If you exchange code with another student, write code while discussing it with a fellow student, or copy code from another person's screen, this code is not yours.
5. **Collaborative coding is strictly prohibited**. **Your assignment submission must be strictly your code**. Discussing anything beyond assignment requirements and ideas is a strictly forbidden form of collaboration. This includes sharing code, discussing the code itself, or modelling code after another student's algorithm. **You can not use (even with citation) another student's code.**
6. Making your code available, even passively, for others to copy or potentially copy is also plagiarism.
7. We will look for plagiarism in all code submissions, possibly using automated software designed for the task. For example, see Measures of Software Similarity (MOSS - https://theory.stanford.edu/~aiken/moss/).
8. Remember, if you are having trouble with an assignment, it is always better to go to your TA and/or instructor for help rather than plagiarizing. A common penalty is an F on a plagiarized assignment.

Generative AI (e.g., ChatGPT)

Refer to the Generative AI policy available in the course outline.

In general, you may use Generative AI to help debug, to learn about a topic or to help develop a strategy to solve a problem. If you use AI, you must cite it (note that you used it) and explain (in your own words) a description of the result.

A Note on AI in Iteration 3

For Iterations 1 & 2, be sure to follow the Guidelines mentioned in the course outline. However, there is a lot to do on iteration 3 of the project and in this case, you are welcome to use Generative AI or other resources (e.g., StackOverflow) as much as you like, providing your team approves. Keep in mind, you still cannot use other team's code and the other plagiarism rules still apply. If you use another source (such as ChatGPT, StackOverflow), be sure to cite your use of it.