

COSC 111: Final Project Overview - Fall 2025

Purpose

The Final Project serves as the culminating experience for COSC 111, allowing you to apply your knowledge of Java programming, problem-solving, and object-oriented design principles to a creative, fully functional application. You will design and implement your own Java program that uses classes, methods, and control structures concepts introduced throughout the semester. This project is divided into multiple phases to help you stay organized and on track.

Project Phases and Deliverables

1. Proposal (Due: November 16)

Submit a short written proposal (½ - 1 page, typed) describing:

- The theme or purpose of your program (e.g., a game, simulation, productivity app, etc.)
- The problem it solves or the user experience it provides
- The features you plan to include
- The classes and methods you expect to use
- Any challenges you anticipate

Goal: Demonstrate initial planning and ensure your project scope is realistic and meets course expectations.

2. Phase 1 - Planning and Initial Implementation (Due: November 23)

Deliverables:

- UML Class Diagram showing the relationships between your classes
- At least one student-defined class created and tested
- Main class started (with pseudocode or partial functionality)
- Submit your `.java` files and UML diagram

Goal: Show your initial structure and class design, ensuring your project is off to a strong start.

3. Phase 2 - Development and GitHub Setup (Due: December 5)

Deliverables:

- GitHub Repository created for your project
 - Repository should include your `.java` source files and a clear `README.md`
- Proof of progress: working version of your code with partial functionality
 - Your program should compile and run, even if incomplete
- Include comments throughout your code explaining each section

Goal: Demonstrate steady progress and proper version control practices.

4. Final Demo and Submission (Due: December 15)

Refer to the Final Project Rubric for grading details. To earn full credit, your final submission should include:

- A fully compiling program that is menu-driven, well-documented, and error-tolerant
- At least one student-defined class and three methods of your own creation
- A clear and readable UML diagram
- A well-formatted GitHub repository with a descriptive `README.md`
- A short reflection document (`.doc` or `.docx`) describing:
 - Your design process and reasoning
 - How you selected algorithms or logic
 - Lessons learned and challenges overcome
- **A 3-5 minute (minimum) video demonstrating your program's functionality including all methods, class definitions, and logic choices. This can be done via**

screen-capturing in Zoom or another screen-sharing tool. There must be audio (of you speaking/demonstrating) and visuals of your code and its functionality.