CSc 335
Spring 2025
Final Project

For the final project in this class, you must complete a large software project in a group of 3 or 4 people. Your project will be graded based on *functionality*, *complexity, collaboration, design*, and *documentation.* What follows in this document are the general expectations for the project based on those four categories. There are also documents provided for specific project ideas, but you do have the opportunity to come up with your own idea if you choose as long as you get it approved and it meets the general requirements.

I. Functionality

This means that you produce working software and evidence that it works as intended. This means that you have to not only submit code that compiles and runs but you have to submit evidence that it runs. This includes

- a suite of unit tests that run, pass, and provide at least 90% coverage for each class in the backend
- a fully functional project where fully functional depends on the expected functionality of your specific project
- evidence that the unit tests pass and that the project is fully functional – meaning you actually run the code and show all the functionality with complex enough test cases

Note that providing evidence is required in addition to submitting the code. How you provide this evidence depends on how you choose to be graded, which will be discussed in a later section.

II. Complexity

There is no line number expectation because often shorter code is better than longer code, so I do not want you writing more code just for the sake of writing more code. Instead, here are some guidelines for gauging the complexity of your project.

- The complexity of the whole project should be about the same as the complexity of LA 1 + LA 2 + the required additional advanced feature that will be explained later.
- The backend code should involve multiple classes interacting with each other.
- The backend code should also utilize Java features and data structures.
- This gets into the design portion, but the backend code should show thoughtful design based on the things discussed in class.
- Each person in the group should put in approximately 40 hours on this project (over the course of the approximately 5 weeks you have to work on it).

In addition to the basic requirements, you must include *at least one* more advanced feature. Although your backend code should mostly be your own code, you may use Generative AI to assist in implementing these more advanced features. (You are also allowed to use Generative AI to implement the user interface code, whether it is a text-based UI or a GUI). Here are some ideas for more advanced features, which will require some research on your end as well as a

thoughtful approach to how it fits with your project. I strongly recommend that you decide on this before deciding fully on a project idea because not all of these fit with all the projects.

- a high-quality graphical user interface that correctly utilizes the OBSERVER design pattern – you can use either Swing or JavaFX for this
- concurrent programming with threads
- network programming – e.g. sending data over sockets
- advanced security features – this would have to go well beyond the simple salting and hashing that was required for LA 2
- use of more advanced design patterns, such as COMPOSITE or DECORATOR (keep in mind that these are context-dependent, and should not be forced)
- metaprogramming

Note: Some of these would likely require you to design a project that fits. If you are interested in one of these advanced features and need help coming up with a project, come see me.

Make sure you include an explanation of whatever additional features you include in your video or grading session.

III. Collaboration

This is a group project, and you are required to work in groups of 3 or 4 people. The preference is for groups of 4. You are strongly encouraged to find your own group, and if you submit a group of 3, please note that you may get another member added. Here are the general expectations for encouraging and assessing collaboration:

- You must email mramoshernandez@arizona.edu with your group members by the deadline listed below. (Note: only one email per group is required.)
- You must use Github to collaborate on your code, and we will check your repo and commit histories to make sure that all members have been contributing to the project the whole time. Procrastination is not acceptable for this project. You also need to make sure you're using Github properly.
- You are expected to utilize some of the principles of Agile development, including using "sprints" to plan manageable tasks that can then be checked.
- You are expected to meet with your grader once in the third or fourth week for a 15-minute "standup" to discuss how the project is going. This can be in person or on Zoom, but every member of the group must participate. Your grader should contact you early in Week 2 to schedule that meeting.
- You will be asked to provide a rating for each of your team members at the end in order to let us know how the collaboration went. Specific guidelines for this are provided in a separate document, and it's important to note that these ratings do affect your grade on the project.

IV. Design

This part is really important. Do not ignore this part! It's not enough for you to just produce working code. You need to provide evidence that you have thought about the design (especially of the backend). Again, how you provide that evidence depends on

how you choose to be graded, which will be discussed later, but you need to be able to explain and justify your design decisions. We will also be looking at your design and your source code, looking for issues in the design. Here are some guidelines for good design based on the course content:

- Clear separation of concerns between front and backend code
    - all UI functionality should be in the View
    - in other words, you should be using MVC in a correct and clear way
- Thoughtful use of data structures and Java library features
    - provide a valid justification for the data structures you use – e.g. if you choose to use ArrayLists for everything, you need to give a valid justification for that decision
    - you also need to show that you have looked at the options in Java and used what is available in a thoughtful way – e.g. if you're writing your own sorting method for a collection, that isn't a good sign when you can easily use the Comparator interface and Collections.sort method to do the same thing
- Correct and thoughtful use of composition, inheritance, and/or interfaces – Your backend code needs to have multiple classes that interact with each other, and you need to be able to explain the design of each class and the choices you made with respect to how they interact.
- Encapsulation –
    - Make sure you have well-encapsulated classes in the backend and can explain how they are well-encapsulated.
    - Avoid escaping references as much as you possibly can. If there are any in your code, make sure you can justify leaving them in.
- Avoidance of antipatterns –
    - We will look for obvious antipatterns in your code.
    - We also want to hear about your process for avoiding them. For example, if you chose to use an enum in order to avoid PRIMITIVE OBSESSION, be prepared to explain that.
- Use of design patterns –
    - These are context-specific, but you should use them when appropriate.
    - We will look for obvious places where you should have used a design pattern and didn't.
    - We will also want to hear about your process for utilizing a design pattern, and we will look to see if you used it appropriately and correctly.
    - Note: Although MVC is sometimes called a design pattern, everyone should be doing this already, so don't focus on that one in this part.
- Input validation – be able to explain how you do any appropriate input validation and justify the type of input validation you choose to use.
- Explain what any AI-generated code does.

V. Documentation

- You need to provide at least one UML diagram showing the overall design of your backend code. This needs to include correct use of the connectors between classes. The UML diagram cannot be handwritten. The purpose of this diagram is to show the interaction of the classes in the backend, so you do not need to include every detail of every class in the diagram.
- Your code should be well-documented using helpful, explanatory comments.
- Provide a README document explaining the design of your code (See Part V for the expectations). We will use this in order to fill in gaps that may be missing in the main grading method as explained below. The README should also explain how to run your code.
- All AI-generated code must be documented, and you need to be prepared to explain what it does.

GRADING.
Here is the general breakdown for points for each category above.

Functionality	40 points
Complexity	25 points
Collaboration	20 points
Design	40 points
Documentation	25 points

Providing Evidence for Functionality and Explaining Design
In addition to the items you need to submit, which are detailed below, you have two options for how to provide evidence for the functionality of your code and for explaining your design.

Option 1 – Provide a video in .mp4 format that is no longer than 25 minutes. Note that this would be a major artifact we use to grade functionality and design, so it will need to be of good quality, well-organized, and very thorough. The graders will not go in and try to run your code themselves in order to fill in the gaps. All members of the group need to be involved.

Option 2 – Make an appointment with your grader (no longer than 30 minutes) to show and explain your project. Particularly, you would need to run the code for them, showing them all the functionality and explaining the design. Make sure you prepare for this as the graders do not have time to spend longer than 30 minutes, so you need to plan and practice. All members of the group must be present and involved.

What to submit (and when).

Item	**Deadline**

Email mramoshernandez@arizona.edu with your group members. Only one email per group is necessary.	Thursday, March 27, by 11:59 PM
Email lotz@cs.arizona.edu with your project idea and some details for how it will meet the general requirements (if you are choosing your own project)	Friday, March 28, by 11:59 PM
Share Github repo with grader and schedule “standup”	Friday, April 4, by 11:59 PM
Meet with grader for “standup”	Friday, April 18, by 11:59 PM
Final submission on D2L – A zipped folder containing: ● source code ● documentation ● video (if using Option 1)	Friday, April 25, by 11:59 PM
Schedule grading session with grader (if using Option 2)	Friday, April 25, by 11:59 PM
Individual: Peer Evaluation on D2L	Friday, April 25, by 11:59 PM

Policy on AI use:

- You may use generative AI for the user interface code.
- You may use generative AI to help with the additional advanced feature(s) explained in Part 2.
- You must document all code that is generated by AI.
- You must be prepared to explain all code that is generated by AI.
- You may not use AI on any other part of the code or the rest of the project.