**OOP and Design Patterns (CSCI 375)**

**Student Showcase (Final Project) Rubric**

**Self-Grade**

**Grading Rubric:**

**Instructions:**

1. There are 9 technical requirements to grade the project and the team presentation.
2. For each requirement, use 0.0 - 5.0 scale in the Score column (0 is Fail/F and 5 is Excellent/A+).
3. Use the *Notes* section to jot down any observations that may help in grading and justification.

| **Team and Technical Project Requirement** | **Score** |
| --- | --- |
| 1. Use of fundamental OOP concepts, e.g.: Inheritance, Abstraction, Attributes, Getters, Setters, Methods, Modularity, Overloading, etc.   **Notes**: Used getters, setters, attributes, methods, inheritance (publisher.py), but doesn’t use more complex OOP concepts (overloading/modularity) | 4/5 |
| 1. Use of at least 3 Design Patterns -- presentation clearly stated and briefly explain design patterns use. Common design patterns are Iterator, Decorator, Observer, Strategy, Command, State, Singleton, Adapter, Façade, Flyweight, Abstract Factory, Composite, Template, MVC, etc.   **Notes**: Comments at beginning of main.py code explain the 3 design patterns implemented (**Observer**, **Singleton**,and **Facade**) and where they are used | 5/5 |
| 1. Unit testing for correctness – automatically generates test data using hypothesis, usage of mocking/patching, provides code coverage and Python type check (mypy) reports, etc.   **Notes**: Used **MyPy**, **unit testing** for testing types (**hypothesis** also briefly used to generate int data, see test\_main.py in repo), manual testing | 4/5 |
| 1. Documentation – clear, easy to follow documentation, UML diagrams are complete, and notations are correct; explanation of objects interaction is clear and complete.   **Notes**: Made **UML** diagrams (are in the uml folder in repo), show what’s in each class and how each class interacts with each other and what is managing what. Also used **Docstrings** and generated html files | 5/5 |
| 1. Software management – good usage of management, communication and tracking tools e.g., Gant chart, Kanban board, GitHub, Clickup, Discord, Slack, etc.   **Notes**: Used shared **GitHub** repo, **Clickup** checklists, and mainly texted for communicating to update each other on progress, tasks, and questions  **Clickup** examples (used to track who’s working on what, what is done and what isn’t, due date reminders to show when important tasks for Showcase need to be done): | 5/5 |
| 1. Teamwork – clear division of labor and progress tracking; helping each other, etc.   **Notes**: Worked together to resolve merge conflicts, used Clickup together to track progress, both judges agree teamwork was well done | 5/5 |
| 1. Project requirements and execution -- clearly stated functional and technical requirements, project adequately challenging for sophomore-junior students; project demo was clear and concise, etc.   **Notes**: Our demo was clear because we showed how the game works and functions, then talked about the code behind it. We went through the project requirements and discussed how we worked on completing them all | 4/5 |
| 1. Team presentation -- all members participated in presentation, used the visual and oral presentation techniques and tools to engage the audience, etc.   **Notes**: Both of us contributed to the presentation by taking turns talking about our game and project, and both judges agree | 5/5 |
| 1. BONUS: Above and beyond – Team went beyond the above list 1-8 e.g., great User Interface, use of Database, real-world application, client delight and interaction, CI/CD, deployment, etc.   **Notes**: Used Pygame and made the application with client delight and user friendliness in mind (since it is a video game meant for entertainment) | 4/5 |
| **Total Score** | **41/40** |

**Judges’ Average Score: 32/40**

**Self-Grade Score: 41/40 (post fix, after updating code and adding more)**