OOP and Design Patterns (CSCI 375)
Student Showcase (Final Project) Rubric

1. Project Title: Bork's Dungeon
2. Team Members: Paul Connett, Parker Stacy
3. Evaluator: Self-grade

Instructions:

1. There are 10 technical requirements (worth 100 points) to grade the project and the team presentation.
2. For each requirement, use 0 - 10 scale in the Score column (0-6: Fail, 7: C (Average), 8: B (Above Average), 9: A (Good), (10: Excellent)
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 **4+1 Views** to explain the software design to the audience. Notes: Included case, physical, logical, development, and process views	10/10
2. **Use of 3 Design Patterns** -- presentation clearly stated and briefly explained design patterns use. Common design patterns are Iterator, Decorator, Observer, Strategy, Command, State, Singleton, Adapter, Façade, Flyweight, Abstract Factory, Composite, Template, MVC, etc. Notes: Singleton pattern- single game object for entire program execution State pattern - game is one big state machine Template method pattern - Potions both have 'use' method that do different things - Both room types have different 'play_room' methods	10/10
3. **Use of Fundamental OOD Concepts-** e.g., Inheritance, Abstraction, Getters and Setters, Overloading, Attributes and Methods, etc. Notes: used Abstraction, overloading, inheritance, attributes, methods, and properties	10/10
4. **Software management** – good usage of management, communication and tracking tools e.g., Gant chart, Kanban board, GitHub Project, Clickup, Discord, Slack, etc. Notes: used GitHub, pushed/pulled/merged from appropriate branches	10 /10

5. **Documentation** – clear, easy to follow documentation, UML diagrams are complete, and notations are correct; explanation of objects interaction is clear and complete. Notes: Created UML and doc folders	10/10
6. **Testing and CI-CD pipeline – automatically generates test data** using hypothesis, usage of mocking/patching, provides code coverage and Python type check (mypy) reports, CI-CD, Docker, etc. Notes: Used mocking/patching, generated test coverage report with makefile, testing is nearly 100%. CI-CD is functional	9/10
7. **Clean code and Modularity** – project uses appropriate software and system design; the code follows best practices (pep8), is well organized and refactored into packages, modules, classes, etc. Notes: Successfully ran make-check-style with no errors	10/10
8. **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:	10/10
9. **Team presentation** -- all members participated in presentation, used the visual and oral presentation techniques and tools to engage audience, etc. Notes:	10/10
10. **Individual Contributions** – briefly explain how and what each member contributed to the successful execution of the project. Notes: Paul: player.py, item.py, enemy.py, mob.py monster.py, test_game.py, test_room.py, umls Parker: room.py, game.py, monster.py, test_player.py,, test_mobs.py, test_potions.py, docker	10 /10
11. **BONUS: Above and beyond** – Team went beyond the above list e.g., great User Interface, use of Database, security, real-world application, real-world client delight and interaction, etc. Notes:	5/10

Total Score Note: Max score can be 110 due to 10 BONUS points.	104 /100