Testing Document

Unit Testing

For our unit testing, we did whitebox testing and incorporated tests of methods for our three main classes: **MuseumPage**, **Comment**, and **Account**. These were the only classes whose methods were tested because the other classes contained mainly getters and setters. A large portion of these unit tests dealt with the gathering of information from the database and adding new information to the database. Furthemore, many of these tests deal with expected invalid activity, and would check to make sure that the changes to the system that would have occurred if the activity were valid did not happen. In total we have 57 total unit tests.

Integration Testing

For our integration testing portion, we specifically conducted user interface testing as well as use case testing. The user interface testing consisted of all group members going through the system and clicking all buttons, selecting all dropdown menu options, and entering various inputs in the system's Graphical User Interface (GUI). The results of our user interface testing showed that no crashes were found and all inputs/buttons yielded expected results.

As for the use case testing, we utilized our user stories and use cases to verify that the system successfully achieved what the customer had requested. In our use case descriptions, the actions of "visit museum page", "create new museum page", "invite new curators", and "create new accounts" were outlined. After our use case tests, we have verified that the system allows the user to perform the actions described in the use case descriptions according to its parameters.

Systems Testing

We evaluated our system by conducting usability testing and specifically used heuristics evaluation from Jakob Nielson's book *Usability and Engineering*.

1. **Visibility of system status**: There are multiple pop-up menus all over the system that updates the user on the status of their desired action.
2. **Match between system and the real world**: The system speaks with phrases/concepts familiar to the user. For instance, the system uses "Visit" on buttons denoted changing the current museum page the user is on.
3. **User control and freedom**: When writing comments users can cancel what they have written. In museum page creation, users can return to a museum page instead of submitting a museum page
4. **Consistency and standards**: The GUI follows a consist format throughout the system
5. **Error prevention**: Series of if statements in the source code prevent invalid input crashes and there are also many pop up menus to assist the user on why an error has occurred
6. **Recognition rather than recall**: All of the functionality on each museum page is reported by text on the buttons of each museum page, so the user will not have to recall what actions he can take.
7. **Flexibility and efficiency of use**: All widgets and labels make each action self-explanatory
8. **Aesthetic and minimalist design**: Dialogues only include information relevant to the user at that point in time and does not overwhelm the user with unnecessary details
9. **Help users recognize, diagnose, and recover from errors**: Error messages list problems clearly and assists the user with how they can be fixed
10. **Help and documentation**: Both documentation and a user manual are available for all users