Poll Types Assurances:

Each of the poll types and subtypes (Poll, PollQuestion, PollResponse, and children of these) are tested using a qunit test script simply called unitTest.py. This tests the creation of each of these classes (regular, fromDict, fromJson, and fromBytes) as well as the successful translation of these classes from these formats (toDict, toJson, and toBytes). It also tests some of the class methods and functionality such as hiding/showing answers in polls, assigning a student answer to a PollResponse, and more.

Professor Assurances:

The professor UI currently only has 1 control-flow-path, which I tested by running the program with various inputs. When using the program correctly, nothing crashes, and I verified the correct poll shows up in the server's output.

Client Assurances:

Server Assurances:

For the server we do not have any unit tests. Instead, we implemented small chunks of code and tested the functionality by running the server and connecting it to the professor and students. Each of the actions that the server can perform was tested with requests made from professors and students. In order to catch errors quickly we wrap all of the incoming data into one of the types found in polltypes.py. This has allowed us to catch errors quickly because if the received data is not correctly formatted it will not conform to the type and python will relay a useful error message.

General Assurances:

To smooth code integration we took advantage of githubs branches features and created pull requests which were reviewed by one or more members of our group. The peer review process helped catch multiple errors and inconsistencies as well as aid in building familiarization with the growing code base. Additionally, we took advantage of the class time last week and our weekly meeting to pair program and delegate individual responsibilities.