Ideal Test Plan:

- Unit Testing-
    - aipreprocessing.py file
        - Testing whether the script connects to the Postgres database properly.
        - Testing queries to see whether or not they give us the data we need for the models.
        - Testing model inputs
            - Again, validating the conversion of SQL queries into the appropriate dataframes with the correct features and dimensions for the model.
        - Testing correctness of Model outputs
            - Confusion matrix and f1 scores (2 * (precisions * recall) / (precision + recall)) will give us an estimate if the model is working properly.
            - Getting average accuracy across all users for each model and making sure the accuracy and f1 score does not greatly vary between users.
            - Manually checking whether the probabilities for videos suggests a pattern in user behavior that lines up with the data itself (that is having a high probability for predicting sports videos when indeed the user watched a substantial amount of sports videos in their history).
- Integration & Systems Testing-

    - Testing Database connections- This was already somewhat covered in unit testing.
        - Ensuring data is properly read in from the database to the script that trains and executes the model.
- Description of Tools Used-
    - Python's unittest framework - For any methods that were explicitly declared we would use the unittest to evaluate their outputs. Example: The stripdays function in the aipreprocessing.py file.
    - Pandas testing framework - This would probably be used in conjunction with the unit test framework to validate whether the dataframes that are being created from SQL queries are producing correct input for the ML model instances.
    - Scikit-Learn confusion matrix/f1_score - As stated above this would be implemented to evaluate model performance and accuracy.
- End User Descriptions-
    - Tech Lead - The tech lead at City News Beat will ultimately take the code we have produced and integrate into their system.
- Performance & Reliability Testing-
    - Load Test
        - Testing model output/model accuracy when the amount of data increases
            - That is, when the number of users, user interactions, and videos increase by multiple orders of magnitude does the model still produce coherent results and are the accuracy rates/f1-scores unchanged.

        - Whether the connection to the database across the python file is slowed down by increasing the number of data points in the database.
    - Regression Test
        - Upon meeting with the tech lead, perhaps before a final acceptance test, making sure none of the core functionality in the software breaks when the structure is changed to fit the client's architecture requirements.
- Acceptance Testing-
    - Meeting with the tech lead of the City News Beat project and ensuring that our code that we have written can integrate properly with their existing architecture. Namely, making sure our script can establish a proper pipeline from their user data to the appropriate probabilistic outputs for each user. We would also explain to them what the outputs mean so that they can accordingly show recommended videos for each user.

Realistic Test Plan:

- Unit Testing-
    - aipreprocessing.py file
        - Testing whether the script connects to the Postgres database properly *(unchanged from ideal).*
        - Testing queries to see whether or not they give us the data we need for the models *(unchanged from ideal).*
        - Testing model inputs

            - Again, validating the conversion of SQL queries into the appropriate dataframes with the correct features and dimensions for the model *(unchanged from ideal).*
        - Testing correctness of Model outputs
            - Confusion matrix and f1 score (2 * (precisions * recall) / (precision + recall)) will give us an estimate if the model is working properly *(unchanged from ideal).*
            - Getting average accuracy across all users for each model and making sure the accuracy and f1 score does not greatly vary between users *(unchanged from ideal).*
            - Manually checking whether the probabilities for videos suggests a pattern in user behavior that lines up with the data itself (that is having a high probability for predicting sports videos when indeed the user watched a substantial amount of sports videos in their history) *(unchanged from ideal).*
- Integration & Systems Testing-
    - Testing Database connections- This was already somewhat covered in unit testing
        - Ensuring the data from the initial excel file is properly inserted into the database *(unchanged from ideal).*
        - Ensuring data is properly read in from the database to the script that trains and executes the model *(unchanged from ideal).*
- Description of Tools Used-

    - Unit Testing in our case we have determined is not as essential for our application as we only have one data pipeline and eventually one output that is produced, at a time. Therefore a lot of our testing can be done manually, through print statements, or through something of the like. Not to mention, the way our code is structured requires that what we pass into it has the correct format (for example, the models require certain dimensionalities to the feature matrices and label arrays). If we had a UI component to our project it would certainly require suite(s) of unit tests.
    - Scikit-Learn confusion matrix/f1_score - *This is unchanged from our ideal plan.* We regard this as the most important testing that will occur as it tests the validity of what the model is outputting.
- End User Descriptions-
    - Tech Lead - The tech lead at City News Beat will ultimately take the code we have produced and integrate into their system *(unchanged from ideal).*
- Performance & Reliability Testing-
    - Load Test
        - While testing the database connections and model outputs when the data volume increases is indeed a vital part of ensuring that our software can scale properly, we do not have the time or facility to generate hundreds or thousands of more made-up data points.
    - Regression Test
        - We would not be able to meet up with the tech lead of the City News Beat before the final code delivery to see if core functionality remains adapting the software to the client's architecture.

- Acceptance Testing-
    - We will still try our best to meet up with our client to discuss the code we hand off to him. In any case, we will thoroughly document our code.