Django Lecture-Attendant-DirEdge Model - Sprint 1:

1. Lecture Model Test:
    a. Equivalence Class:
        i. Acceptable input = list of non-null ASCII strings with maximum length of twenty characters
        ii. Unacceptable input = non-ASCII input, null input, input over twenty in length
    b. Python Django Shell Test:
        i. Import model
        ii. Save instance of Lecture with acceptable input
            1. lecture_title = “<acceptable input>”
        iii. Query Database to find Lecture instance
        iv. Verification:
            1. Instance represented as <Lecture_title>: <year created>
            2. Lecture_title matches input
            3. pub_date matches time of creation
        v. Wipe out all created instances
2. Attendant Model Test
    a. Equivalence Class:
        i. Acceptable input = list of non-null ASCII strings with maximum length of twenty characters
        ii. Unacceptable input = non-ASCII input, null input, input over twenty in length
    b. Python Django Shell Test:
        i. Import model
        ii. Save instance of Attendant with acceptable input
            1. student_id = “<acceptable input>”
        iii. Query Database to find Attendant instance
        iv. Verification:
            1. Instance represented as <student_id>:
            2. student_id matches input
            3. pub_date matches time of creation
        v. Wipe out all created instances
3. Model one to many integration test:
    a. Python Django Shell Test:
        i. Import Lecture model
        ii. Save instance of lecture model
        iii. Add several Attendants to instance of lecture model
        iv. Query Database to find representation
        v. Verification:
            1. Instance of lecture exists
            2. Attendants associated to lecture instance
        vi. Wipe out all created instances

Student-Student edge Function - Sprint 1:

1. Function Test:
    a. Equivalence Class:
        i. Acceptable input = valid lecture model instance, student_id of attendant object contained in lecture instance, temp_id of separate attendant object, neither attendant object contains a directed edge to each other
        ii. Unacceptable input = attendant objects referenced have a directed edge referring to each other.
    b. Acceptable input test:
        i. Python Django Shell Test:
            1. Import function from views
            2. Import lecture and attendant models
            3. Craft a lecture instance and populate it with two attendant objects
            4. Call function on objects
            5. Verification:
                a. Returns “Connection Success”
                b. Query Database to find a DirEdge in attendant object that provided the temp_id
                c. Connections field in both attendant objects should raise by one
            6. Wipe out all created instances

Attendance List Sorting function:

1. Function Test:
    c. Equivalence Class:
        i. Acceptable input = valid lecture model
    d. Acceptable input test:
        i. Python Django Shell Test:
            1. Import function from views
            2. Import lecture and attendant models
            3. Craft a lecture instance and populate it with three or more attendant objects
            4. Change connections values in attendants so that at least one is -1, one is >-1 and < 2 and one is >=2.
            5. Run function with lecture object
            6. Verification:
                a. Returns a list of attendant lists
                b. [0] contains attendant objects with connections = -1
                c. [1] contains attendant objects with -1 < connections <2
                d. [2] contains attendant objects with connections >= 2
            7. Wipe out all created instances

Django Course- Student-Lecture Model - Sprint 2:

1. Course Model Test:
    b. Equivalence Class:

i. Acceptable input = list of non-null ASCII strings with maximum length of twenty characters
ii. Unacceptable input = non-ASCII input, null input, input over twenty in length

c. Python Django Shell Test:
i. Import model
ii. Save instance of Course with acceptable input
1. course_title = “<acceptable input>”
iii. Query Database to find Course instance
iv. Verification:
1. Instance represented as <course_title>
2. course_title matches input
3. pub_date matches time of creation
v. Wipe out all created instances

4. Student Model Test
a. Equivalence Class:
i. Acceptable input = list of non-null ASCII strings with maximum length of twenty characters
ii. Unacceptable input = non-ASCII input, null input, input over twenty in length

b. Python Django Shell Test:
i. Import model
ii. Save instance of Student with acceptable input
1. student_id = “<acceptable input>”
iii. Query Database to find Student instance
iv. Verification:
1. Instance represented as <student_id>:
2. student_id matches input
3. pub_date matches time of creation
v. Wipe out all created instances

5. Model one to many integration test:
a. Python Django Shell Test:
i. Import Course model
ii. Save instance of Course model
iii. Add several Student objects to instance of course model
iv. Add several Lecture objects to instance of course model
v. Query Database to find representation
vi. Verification:
1. Instance of Course exists
2. Students associated to course instance
3. Lectures associated to course instance
vii. Wipe out all created instances

Populate new lectures with attendants corresponding - Sprint 2:

1. Function Test:
    e. Equivalence Class:
        i. Acceptable input = course_id from valid Course instance
        ii. Acceptable input = list of non-null ASCII strings with maximum length of twenty characters
    f. Acceptable input test:
        i. Python Django Shell Test:
            1. Import function from views
            2. Import Course model
            3. Craft a Course instance and populate it with three or more student objects
            4. Run function with coure_id from course object and <acceptable input ii>
            5. Verification:
                a. Check to see lecture object with lecture_title = <acceptable input ii> associated with Course object
                b. Lecture object should have an attendant object for every student in the course with matching student_id's
            6. Wipe out all created instances