```
"""
Request functions available to student users.
"""
import datetime


class StudentRequestHandler:
    def __init__(self, postgres_handler):
        self.postgres_handler = postgres_handler

    def update_location(self, username, major, minor):
        """
        Allows students to update their location based on the major and minor keys of
        the beacon closest to them. The JSON request accepts:

            username: String
            Major: String (int8)
            Minor: String (int8)

        :returns JSON response with format

            successful: Boolean
            reason: String
            closest_beacon: String
            beacon_description: String
        """

        # Surround with try/except in case of connection errors
        try:
            # The major and minor values first have to be matched to a beacon
            # Create SQL Query string
            sql = "SELECT * FROM beacon WHERE major_key = (%s) AND minor_key = (%s);"
            args = (major, minor,)
            # Beacon received
            beacon = self.postgres_handler.query(sql, args)[0]

            # Then, the student_id value must be obtained.
            # Create SQL Query string
            sql = "SELECT * FROM student WHERE email = (%s);"
            # Format item to be searched for
            username = (username,)
            # Student received
            student = self.postgres_handler.query(sql, username)[0]
            # Student_id obtained
            student_id = student[0]

```

```
47              # Lastly, the location must be added to the database with the
    student,
48              # beacon id, and a server time timestamp.
49              # Create time stamp for current time and convert it to a
    string.
50              timestamp = str(datetime.datetime.now())
51              # Create SQL Query string
52              sql = "INSERT INTO location (location_id , time_stamp,
    closest_beacon_id, student_student_id) VALUES (DEFAULT, %s, %s, %s)"
53              args = (timestamp, beacon[0], student_id)
54              self.postgres_handler.insert(sql, args)
55
56              return {
57                  'successful': True,
58                  'reason': 'success',
59                  # Pass the 'closest_beacon' item the beacon room number
60                  'closest_beacon': beacon[1],
61                  'beacon_description': beacon[2]
62              }
63          except:
64              print("Failed to update location")
65
66
67      def get_info(self, username):
68          """
69          Allows students to get information about themselves to be
    displayed on
70          their screen. The JSON request accepts only one variable:
71
72              username: String
73
74          :returns JSON response with format
75
76              successful: Boolean
77              reason: String
78              name: String
79              email: email
80          """
81
82          # We know that the username exists in the database, because it
    has already been
83          # used to login. So, we can simply make a request without worries
     of errors.
84
85          # We still surround it with a try/except in case of connection
    errors.
86          try:
87              # Create SQL Query string
88              sql = "SELECT * FROM student WHERE email = (%s);"
```

```
 89             # Format item to be searched for
 90             username = (username,)
 91             student = self.postgres_handler.query(sql, username)[0]
 92 
 93             return {
 94                 'successful': True,
 95                 'reason': 'success',
 96                 'name': student[1],
 97                 'email': student[3]
 98             }
 99         except:
100             # Fail condition: Broad fail condition for a connection to
    database error.
101             return {
102                 'successful': False,
103                 'reason': 'Connection Error.',
104                 'name': 'error',
105                 'email': 'error',
106             }
107 
```