```
"""
Used to handle server side login operations.

    login:
        Called with a connection to bottleserver/login
        Used to determine if a login is valid
    check_login:
        Called to test a login using a username and a hashed
        password
    create_account:
        Called when a new user would like to claim a username
"""

import security.password as phandler


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

    def login(self, username, password):
        """
        Called when a user goes to the /login directory of the bottle server.
        Accepts a JSON formatted request file in the form of:

            type: String ('LOGIN')
            email: String
            password: String

        """
        # Hash the password
        hashed_password = phandler.hash_password(password)
        return self.check_login(username, hashed_password)

    def check_login(self, username, hashed_password):
        """
        Selects correct password hash from sql database and compares them using a
        password_handler.py function.

            username: String
            hashed_password: String

        :returns JSON with format

            successful: boolean
            reason: String
        """
```

```
48
49          return_request = {
50              'successful': False,
51              'classification': 'unknown',
52              'reason': 'Unknown'
53          }
54
55          try:
56              # Select correct password from database based on username
57              # Create SQL Queries
58              sql_student = "SELECT password FROM student WHERE LOWER(email
    ) = LOWER('%s');" % username
59              sql_admin = "SELECT password FROM admin WHERE LOWER(email) =
    LOWER('%s');" % username
60
61              # Get the correct passwords for students and administrators
62              correct_password_student = self.postgres_handler.select(
    sql_student)
63              correct_password_admin = self.postgres_handler.select(
    sql_admin)
64
65              # If this is an admin (if it exists in the admin table)
66              if correct_password_admin:
67                  # If the account is from an admin set the
    correct_password to admin's password
68                  correct_password = correct_password_admin
69                  return_request['classification'] = 'admin'
70              # Otherwise, it is a student (it is in the student table)
71              else:
72                  # If the account is from an student set the
    correct_password to students's password
73                  correct_password = correct_password_student
74                  return_request['classification'] = 'student'
75
76              # If the username exists (if the list is not empty)
77              if not len(correct_password) == 0:
78                  # Are the passwords the same? If they are, return that it
     was successful
79                  return_request['successful'] = phandler.compare_passwords
    (correct_password[0][0], hashed_password)
80
81                  # Reason to be printed to user in case of failed login.
82                  if return_request['successful']:
83                      return_request['reason'] = 'Correct login.'
84                  else:
85                      return_request['reason'] = 'inc_login'
86              # Otherwise, the username does not exist
87              else:
88                  return_request['reason'] = 'inc_login'
```

```
 89
 90             return return_request
 91         except:
 92             # Fail condition: Broad fail condition for failure to
    connect to database
 93             return_request['reason'] = 'con_error'
 94             return return_request
 95
 96     def create_account(self, name, username, password, conf_password):
 97         """
 98         Allows users to claim a username that is not in use. The JSON
    request
 99         accepted is in the format:
100
101             name: String
102             email: String
103             password: String
104             conf_password: String
105
106         :returns JSON with format
107
108             successful: boolean
109             reason: String
110
111         ** Makes users login again after claiming account. This is
    because this
112         function only adds to the database, it does not perform the
    login function
113         """
114
115         # Instantiating return_request to be sent to client of server
116         return_request = {
117             'successful': False,
118             'reason': 'Unknown'
119         }
120         # If passwords match
121         if password == conf_password:
122             sql = "SELECT * FROM student WHERE LOWER(email) = LOWER(%s
    );"
123             _username = (username, )
124             # If the email is not in use (not in the database)
125             if not self.postgres_handler.query(sql, _username):
126                 try:
127                     # Hashing password using hashlib
128                     pw_hash = phandler.hash_password(password)
129                     # Adding new user to database: name, username,
    hashed password
130                     sql = "INSERT INTO student (student_id, name,
    password, email) VALUES (DEFAULT , %s, %s, %s)"
```

```
131                     args = (name, pw_hash, username)
132                     self.postgres_handler.insert(sql, args)
133 
134                     return_request['successful'] = True
135                     return_request['reason'] = 'Please login again using
     your credentials.'
136                     return return_request
137                 except:
138                     # Fail condition: Broad fail condition for failure
    to connect to database
139                     return_request['reason'] = 'con_error'
140                     return return_request
141             else:
142                 # Fail condition: Email (username) in use
143                 return_request['reason'] = 'email_use'
144                 return return_request
145         else:
146             # Fail condition: Passwords do not match
147             return_request['reason'] = 'pass_match'
148             return return_request
149 
```