Login Backend

Using

REST API

Abhinav Gorantla

19BCE0241

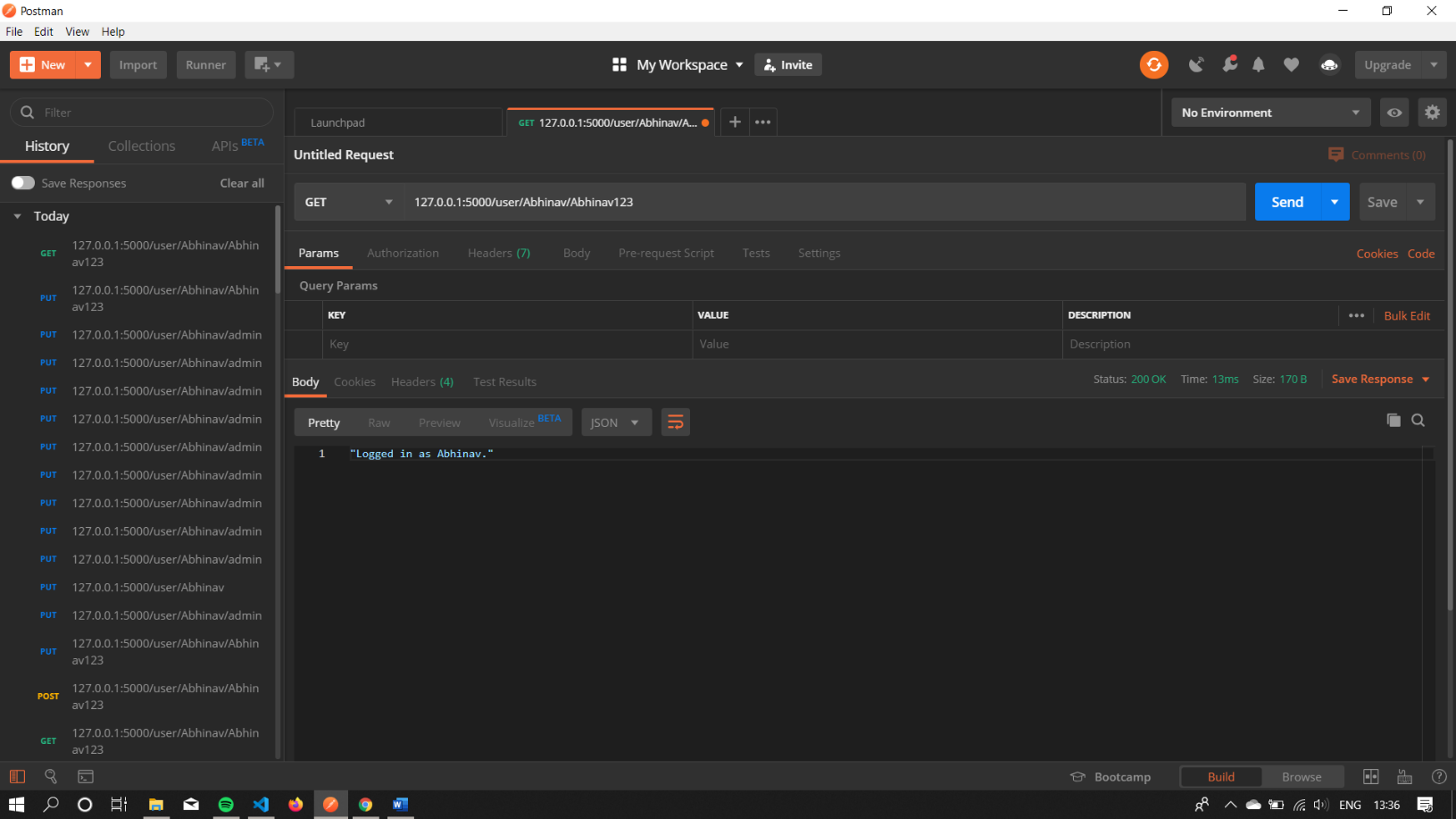
Features:

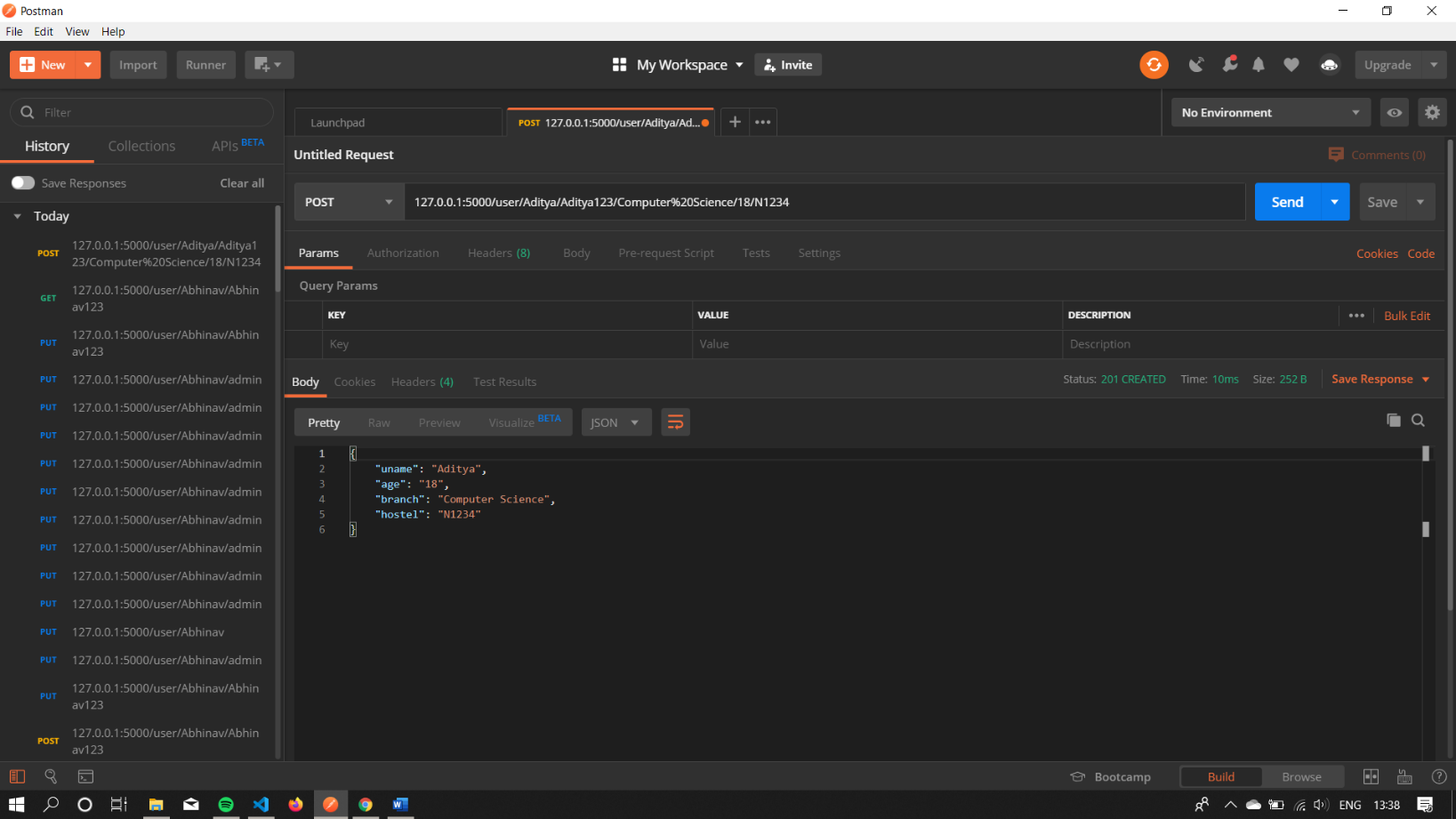
1. The API is designed keeping in mind the clubs and chapters of VIT who would want to create a login interface for their members.
2. This login API contains GET, POST, PUT and DELETE methods.
3. It allows to add users to the existing set of users.
4. Users can also change their details like their age, branch and hostel block.

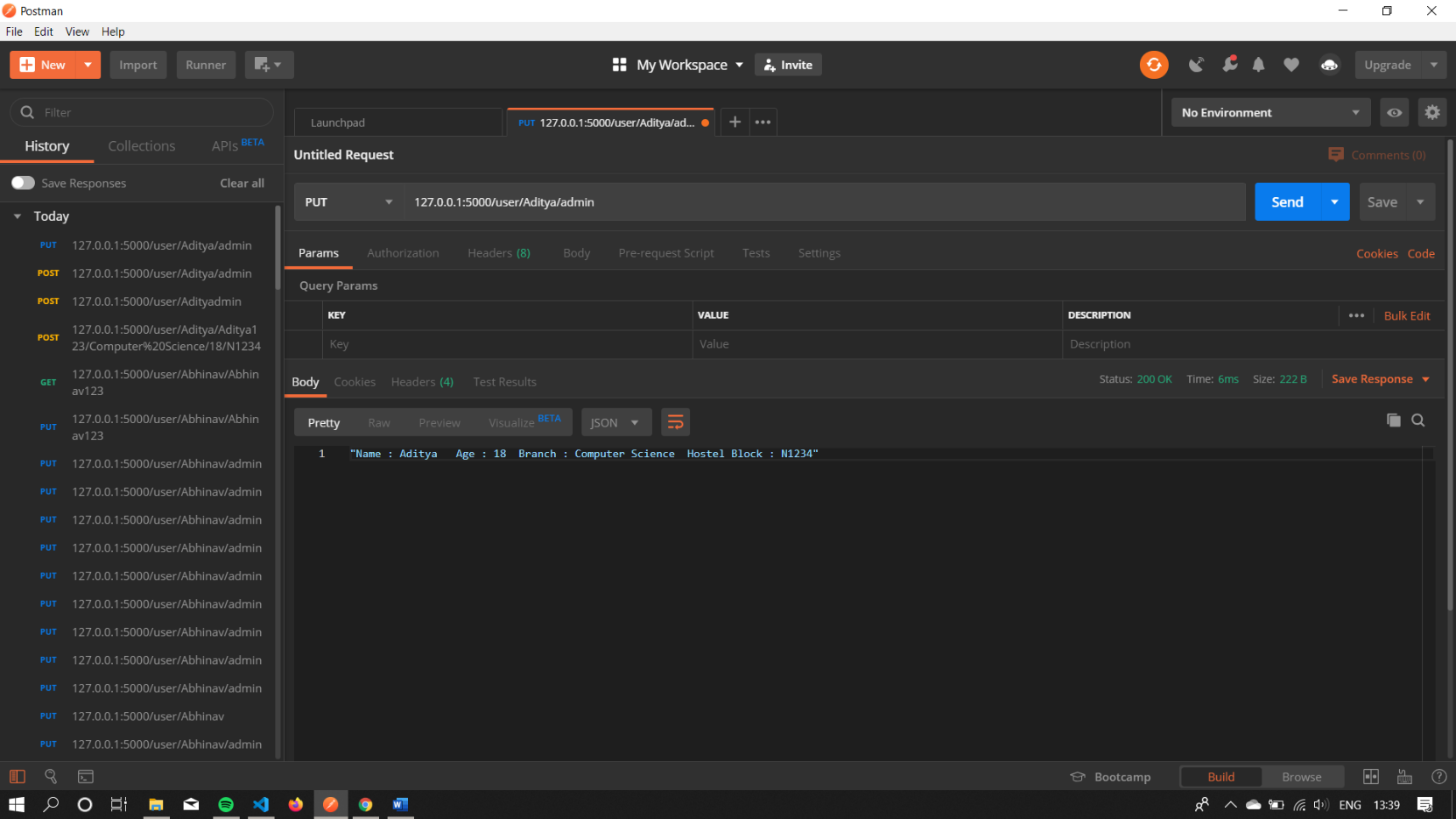
Execution of the project:

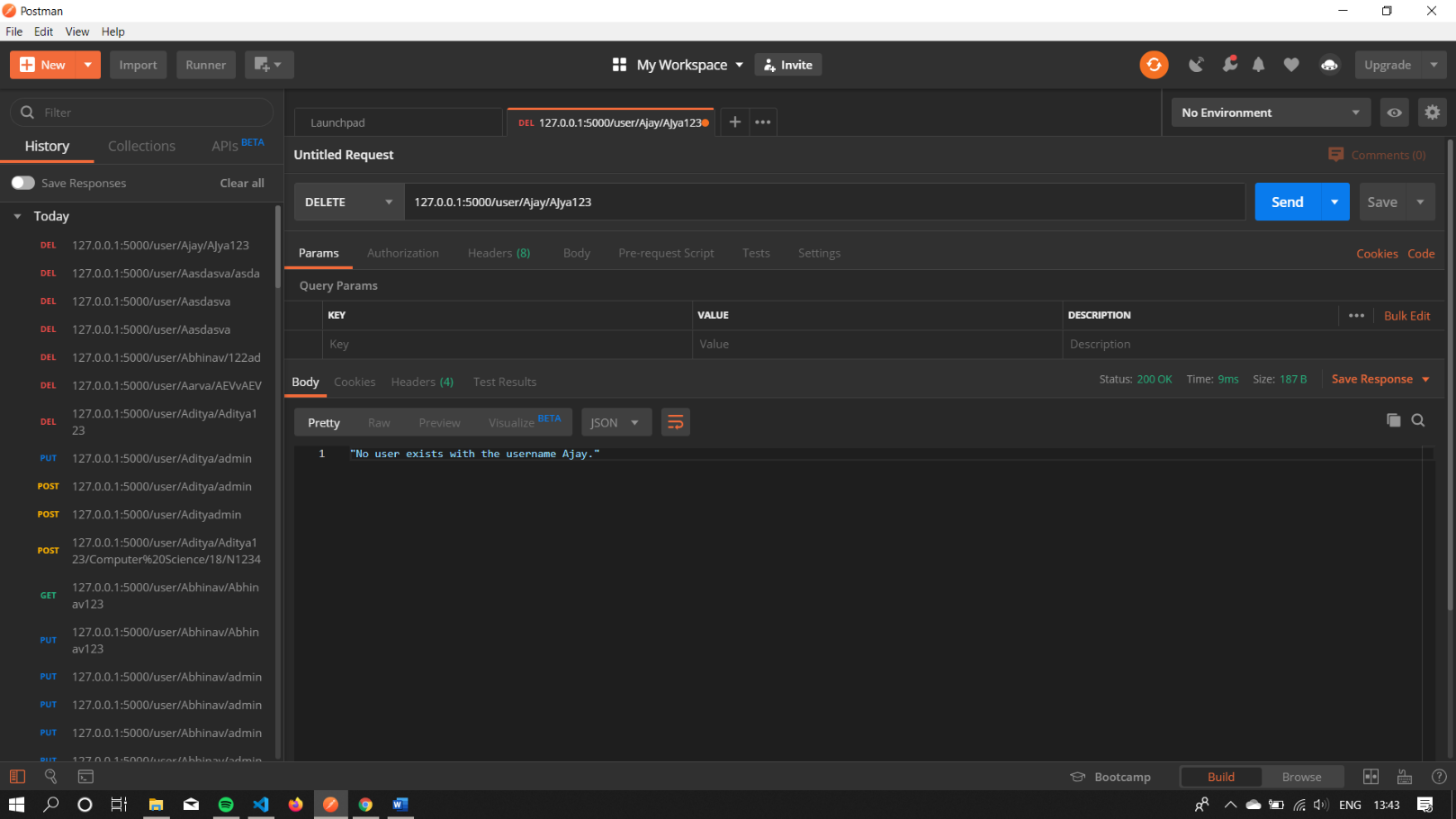
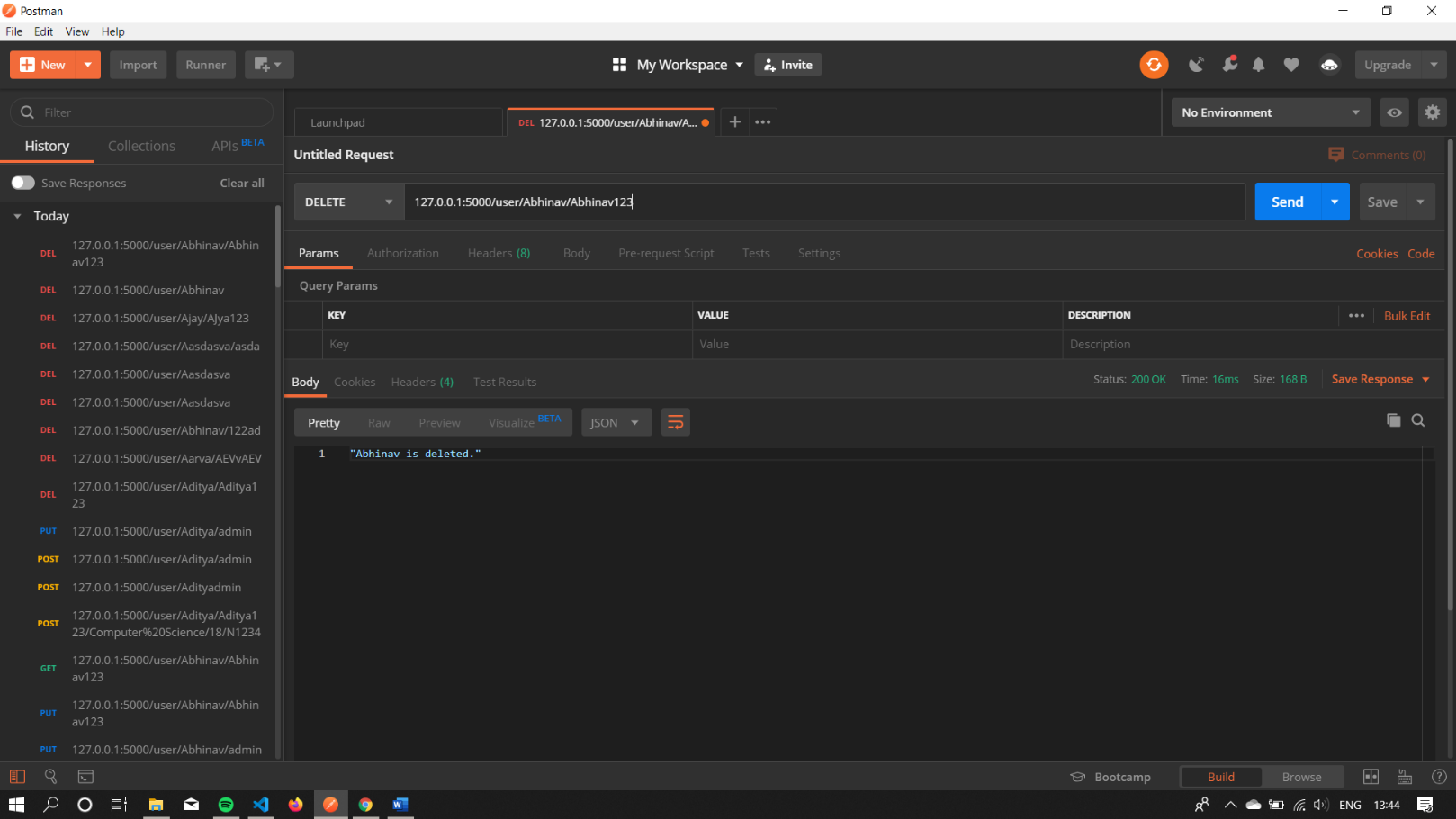
1. *Code used:*
2. from flask import Flask
3. from flask\_restful import Api, Resource, reqparse
4. app = Flask(\_\_name\_\_)
5. api = Api(app)
6. users = [
7. {
8. "uname": "Abhinav",
9. "age": 18,
10. "password" : "Abhinav123",
11. "branch" : "Computer Science",
12. "hostel" : "M111"
13. },
14. {
15. "uname": "Bavesh",
16. "age": 18,
17. "password" : "Bavesh123",
18. "branch" : "Computer Science",
19. "hostel" : "M112"
20. },
21. {
22. "uname": "Jeethendra",
23. "age": 18,
24. "password" : "Jeethendra123",
25. "branch" : "Civil",
26. "hostel" : "M113"
27. },
28. {
29. "uname" : "Chaitanya",
30. "age" : 18,
31. "password" : "Chaitanya123",
32. "branch" : "Aeronutical",
33. "hostel" : "M114"
34. }
35. ]
36. class User(Resource):
37. # This method is used to login
38. def get(self, uname,password):
39. for user in users:
40. if(uname == user["uname"] and password == user["password"]):
41. return "Logged in as {}.".format(user["uname"]), 200
42. elif(uname == user["uname"] and password != user["password"]):
43. return "Password incorrect."
44. return "User not found", 404
45. #Used to add a new user
46. def post(self, uname, password, age, branch, hostel):
47. parser = reqparse.RequestParser()
48. parser.add\_argument("age")
49. parser.add\_argument("branch")
50. parser.add\_argument("password")
51. parser.add\_argument("hostel")
52. for user in users:
53. if(uname == user["uname"]):
54. return "User with name {} already exists".format(uname), 400
55. user = {
56. "uname": uname,
57. "age": age,
58. "branch": branch,
59. "hostel" : hostel,
60. }
61. users.append(user)
62. return user, 201
63. #Used to get the details of a user
64. def put(self, uname):
65. parser = reqparse.RequestParser()
66. parser.add\_argument("age")
67. parser.add\_argument("branch")
68. parser.add\_argument("password")
69. parser.add\_argument("hostel")
70. for user in users:
71. if(uname == user["uname"]):
72. return ("""Name : {}   Age : {}  Branch : {}  Hostel Block : {}""".format(str(user["uname"]), str(user["age"]), str(user["branch"]), str(user["hostel"]))), 200

75. #Used to delete an existing user.
76. def delete(self, uname, password):
77. global users
78. for user in users:
79. if uname == user["uname"]:
80. users = [user for user in users if user["uname"] != uname]
81. return "{} is deleted.".format(uname), 200
82. else:
83. return "No user exists with the username {}.".format(uname)
84. api.add\_resource(User, "/user/<string:uname>/<string:password>/<string:branch>/<string:age>/<string:hostel>","/user/<string:uname>/<string:password>/<string:branch>/<string:age>", "/user/<string:uname>/<string:password>", "/user/<string:uname>/admin")
85. app.run(debug=True)
86. *Prototyping using Postman:*
    1. GET Method:



* 1. POST Method:
  2. PUT Method:



* 1. DELETE Method:
     1. If username doesn’t exist:
     2. If username exists

1. *Things I can improve:*
   1. I need to find a way to hide sensitive data like password from the URL.
   2. Admin authentication for deleting a user needs to be added.