## 1) LOGIN

- URL: http://127.0.0.1:8000/api/login/
- Request: POST
- Input: email and password in the below format
- Json = {

```
    "email": "abc@xyz.pk",
    "password": "******"
}
```

- output: id, state and message in the below format
- CONDITION 1: (If user/password are wrong) – Display the error message “wrong email/pass”
- Response = {

```
    "state": 0,
    "message": "Wrong Username/Password"
}
```

- CONDITION 2: (If Account is disabled) – Display the error message “account disabled”
- Response = {

```
    "state": -1,
    "message": "Account has been disabled by an admin"
}
```

- CONDITION 3: (login success – Ask for height and weight). If message “register” is received then redirect user to height and weight form.
- Response = {

```
    "id": "mem1",
    "state": 1,
    "message": "register"
}
```

- CONDITION 4: (login success – Go to Home Screen). If message “login” is received then redirect user to home screen.
- Response = {

```
    "id": "mem1",
    "state": 1,
    "message": "login"
}
```

## 2) HEIGHT AND WEIGHT FORM

- URL: http://127.0.0.1:8000/api/putmeasurements/

- Request: POST

- Input: id, height and weight. Height and weight must be numbers but converted to string and height variable must contain two number separated by a COMMA

- Json = {

```
    "id": "mem1",
    "height": "5,9",
    "weight": "77"
}
```

- output: state and message variables in below format

- CONDITION 1: (If user id is wrong)
- Response = {

```
      "message": "can't locate user"
}
```

- CONDITION 2: (user exists but updation fails) – state variable will be returned with Value 0
- Response = {

```
    "state": 0,
}
```

- CONDITION 3: (user exists and updation successfully occurs) – state variable will be with Value 1
- Response = {

```
    "state": 1,
}
```

## 3) HOMESCREEN

- URL: http://127.0.0.1:8000/api/homescreen/

- Request: POST

- Input: id and day of the week eg: “monday” etc. for fetching diet and workout routine for that specific day along with user details.

- Json = {

```
    "id": "mem1",
    "day": "monday",
}
```

- output: state and user information in below format

- CONDITION 1: (If user id is wrong)
- Response = {

```
        "message": "can’t locate user"
}
```

- CONDITION 2: (user is located) - then a response will be returned containing user information and routine/diet data for that specific date.

  **NOTE: if user is not following any routine or diet then value 0 bill be returned instead, example “routine”: 0 or “diet”: 0 like below.**

- Response = {

```
        "id": "mem1",
        "name": "Mr. XYZ",
        "contact": "03********1",
        "email": "email",
        "address": "user address",
        "dob": "1994-10-02",
        "height": "10,10",
        "weight": "77",
        "image": "base64 string",
        "routine": 0, //if user is not following any routines
        "diet: 0 //if user is not following any routines
}
```

**Otherwise Diet and Routine Data will also be returned along with user information in the below format.**

- Response = {


```
"id": "mem1",
"name": "Mr. XYZ",
"contact": "03********1",
"email": "email",
"address": "user address",
"dob": "1994-10-02",
"height": "10,10",
"weight": "77",
"image": "base64 string",
"routine": {

        "ex1": {
                "exercise": "exr1",
                "sets": "3",
                "reps": "12",
                "notes": "a description about exercise",
                "name": "Bench Press",
                "image": "base64 image of exercise"
        },
        ……
        ……

        "exn": {
                "exercise": "exrn",
                "sets": "3",
                "reps": "12",
                "notes": "a description about nth exercise",
                "name": "nth exercise name",
                "image": "base64 image of nth exercise"
        }
},

"diet": {

        "Meal 1": {
                "name": "Meal 1",
                "meal": "3 boiled eggs and blab la bla",
                "time": "09:00"
        },
        ……
        ……

        "Meal n": {
                "name": "Meal n",
                "meal": "nth meal description",
                "time": "14:00"
        }

}
```


}

**The diet and routine data are just for that specific day that is mentioned in the request in json. If no routine or diet exists for that day then 0 will be returned**

## 4) USER DIET

- URL: http://127.0.0.1:8000/api/userdiet/
- Request: POST
- Input: id of user
- Json = {

```
    "id": "mem1"
}
```

- output: JSON of user diet for all days Monday - Sunday

- CONDITION 1: (If user is not following any diet)
- Response = {

```
        "diet": 0
}
```

- CONDITION 2: (if user is following a diet) – then a json with following objects will be returned
- Response = {

```
        "monday": {

                        "Meal 1": {
                                "name": "Meal 1",
                                "meal": "3 boiled eggs and blab la bla",
                                "time": "09:00"
                        },
                        ……
                        ……

                        "Meal n": {
                                "name": "Meal n",
                                "meal": "nth meal description for monday",
                                "time": "14:00"
                        }

        },
        …. Rest of the days here just like above with keys "lowercase day name"
        "sunday": {

                        // Sunday meal details in above format
        }

}
```

## 5) USER ROUTINE

- URL: http://127.0.0.1:8000/api/userroutine/
- Request: POST
- Input: id of user
- Json = {

```
    "id": "mem1"
}
```

- output: JSON of user routine for all days Monday - Sunday
- CONDITION 1: (If user is not following any routine)
- Response = {

```
    "diet": 0
}
```

- CONDITION 2: (if user is following a routine) – then a json with following objects will be returned
- Response = {

```
    "monday": {
                "ex1": {
                    "exercise": "exr1",
                    "sets": "3",
                    "reps": "12",
                    "notes": "a description about exercise",
                    "name": "Bench Press",
                    "image": "base64 image of exercise"
                },
                ……
                ……
                "exn": {
                    "exercise": "exrn",
                    "sets": "3",
                    "reps": "12",
                    "notes": "description about nth exercise for monday",
                    "name": "nth exercise name",
                    "image": "base64 image of nth exercise"
                }
    },
    …. Rest of the days here just like above with keys "lowercase day name"
    "sunday": {
                // Sunday exercise details in above format
    }

}
```

## 6) IDENTIFY MACHINE

- URL: http://127.0.0.1:8000/api/identifymachine/
- Request: POST
- Input: base64 string of machine
- Json = {

```
    "image": "base64 string of machine"
}
```

- output: JSON of Exercise and its tutorial video

- CONDITION 1: (if no exercise found)
- Response = {

```
    "message": "no exercises found"
}
```

- CONDITION 2: (if machine is identified) – following response will be returned. Could be a data of single exercise or could be data of multiple exercises associated with that machine and data will be provided in the below format.

- Response = {

```
    "Exercise1": {
        "name": "Bench Press",
        "equipment": "Bench",
        "video": "embed youtube link here"
    },
       …….
       ……. rest of machines data

    "Exercisen": {
        "name": "Nth exercise name",
        "equipment": "butterfly",
        "video": "embed youtube link here for nth exercise"
    }
}
```

## 7) LIST OF ROUTINES AT GYM

- URL: http://127.0.0.1:8000/api/listofroutines/
- Request: POST
- Input: NOTHING/EMPTY

- output: JSON of Routines or a message

- CONDITION 1: (if no routines found)
- Response = {

```
    "message": "no routines found"
}
```

- CONDITION 2: (routines found) – list of routines will be returned in the format below
- Response = {

```
    "Routine1": {
            "id": "rtn1",
            "name": "Fat Burner",
            "description": "description of the routine",
            "image": "base64"
    },

    …

    …

    "RoutineN": {
            "id": "rtnN",
            "name": "nth routine name",
            "description": "description of the NTH routine",
            "image": "base64"
    }

}
```

**The above data must be displayed in a list form to the user with a follow view button and if user clicks on the view button then another post request will be made to the endpoint described in the following page and you must provide the routine id to get the content of that routine.**

## 8) VIEW ROUTINE

- URL: http://127.0.0.1:8000/api/viewroutine/
- Request: POST
- Input: routine id
- Json = {

```
    "routine": "rtn1"
}
```

- output: JSON of Routine of this specific id
- Response = {

```
        "id": "rtn1",
        "name": "Fat Burner",
        "description": "description of the routine",
        "image": "base64 of routine”
         "routine": {
                    //Routine information from Mon - Sun just like above in
                     user routine view
          }

}
```

**A follow button will appear at top of this screen and if user clicks on this follow button then user's current routine will change to this routine, he/she just followed from the list. After clicking on follow button, a request will be made to the endpoint mentioned in the following page.**

## 9) Follow ROUTINE

- URL: http://127.0.0.1:8000/api/followroutine/

- Request: POST

- Input: user id and routine id

- Json = {

```
        "id": "mem1",
        "routine": "rtn2"
}
```

- output: JSON in the below format

- CONDITION 1: (if updation fails) – state variable will be returned with Value 0
- Response = {

```
    "state": 0,
}
```

- CONDITION 2: (if user successfully follows) – state variable will be with Value 1
- Response = {

```
    "state": 1,
}
```

## 10 A) SETTINGS – Updating User Image

- URL: http://127.0.0.1:8000/api/settings/
- Request: POST
- Input: user id, base64 image and update variable that must be set to image
- Json = {

```
        "update": "image", //MUST BE SET TO image
        "id": "mem1",
        "image": "base64"
}
```

- output: JSON in the below format

- CONDITION 1: (if image updation fails) – state variable will be returned with Value 0
- Response = {

```
    "state": 0,
}
```

- CONDITION 2: (if image updation occurs) – state variable will be with Value 1
- Response = {

```
    "state": 1,
}
```

**Update variable must be set to image for uploading an image always. Image validation must be performed before sending a request. Object must be an image**

## 10 B) SETTINGS – Updating Password

- URL: http://127.0.0.1:8000/api/settings/
- Request: POST
- Input: user id, new password, old password and update variable set to password
- Json = {

```
"update": "password", //MUST BE SET TO password
"id": "mem1",
"oldpassword": "userpassword",
"newpassword": "newuserpassword" // NEW PASSWORD MUST BE >= 6 char length
```

}

- output: JSON in the below format

- CONDITION 1: (if password updation fails) – state variable will be returned with Value 0
- Response = {

```
"state": 0,
```

}

- CONDITION 2: (if password updation successfully occurs) – state variable will be with Value 1
- Response = {

```
"state": 1,
```

}

- CONDITION 3: (wrong old password typed) – error message will be returned
- Response = {

```
"message": "type old password correctly”
```

}

**Update variable must be set to password for updating user password always.Old password validation must be performed and it should be of length > = 6 characters.**

## 10 C) SETTINGS – Updating User Data

- URL: http://127.0.0.1:8000/api/settings/
- Request: POST
- Input: user id, name, contact, dob, address, height, weight and update variable that must be set to data.
- Json = {

```
    "update": "data",
    "id": "mem1",
    "name": "mr xyz",
    "contact": "03031234567",
    "dob": "YYYY-MM-DD", //must be in the following format in a string
    "address": "XYZ STREET KARACHI",
    "height": "5,10",      //must be two numbers separated by comma
    "weight": "79"
}
```

- output: JSON in the below format

- CONDITION 1: (if data updation fails) – state variable will be returned with Value 0
- Response = {

```
    "state": 0,
}
```

- CONDITION 2: (if data updation successfully occurs) – state variable will be with Value 1
- Response = {

```
    "state": 1,
}
```

- CONDITION 3: (wrong userid sent) – error message will be returned
- Response = {

```
    "message": "can't locate user"

}
```

**Update variable must be set to data for updating user data always. And date of birth must be a string in the YYYY-MM-DD format. Height must also be a string with two numbers separated by a comma. Weight must also be in a comma. And all the values must be sent even if user changes only a single variable. All the previous values for other variables to be sent through json just in the above format for input.**