

## Unit testing

### 1.Register user

(Create op.)

Successful registration of new user

The screenshot shows a REST client interface with the URL `http://127.0.0.1:8010/register_user`. The method is **POST**. The body is set to **JSON** and contains the following request body:

```
1 {
2   "name": "loganayakic",
3   "email": "loganayakic@gmail.com",
4   "pwd": "logac"
5 }
```

The response is displayed in the **Body** tab, showing a **200 OK** status with a time of 350 ms and a size of 275 B. The response body is:

```
1 {
2   "status": "success",
3   "data": "Data Added"
4 }
```

### 2. Trying to register already existing user based on email id.

o/p: Data already available

The screenshot shows the same REST client interface with the URL `http://127.0.0.1:8010/register_user`. The method is **POST**. The body is set to **JSON** and contains the following request body:

```
1 {
2   "name": "loganayaki",
3   "email": "loganayaki@gmail.com",
4   "pwd": "loga"
5 }
```

The response is displayed in the **Body** tab, showing a **200 OK** status with a time of 7.07 s and a size of 286 B. The response body is:

```
1 {
2   "status": "Failed",
3   "data": "Data already Available"
4 }
```

### 3. Adding contact for registered user

User id must be registered user id  
i/p: Name, email and phone no.  
o/p: Contact Added

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:8010/add_contact`. The request body is a JSON object: `{ "user_id": "venket@gmail.com", "name": "sangee", "email": "sangeetha@gmail.com", "phno": "7878934890" }`. The response is a 200 OK status with a JSON body: `{ "status": "success", "data": "Contact Added" }`.

```
1 {
2   "user_id": "venket@gmail.com",
3   "name": "sangee",
4   "email": "sangeetha@gmail.com",
5   "phno": "7878934890"
6 }
```

```
1 {
2   "status": "success",
3   "data": "Contact Added"
4 }
```

### 4. Fetch all contact of a particular user.

i/p: user id  
o/p: list of contacts

(Read op.)

The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:8010/all?email=venket@gmail.com`. The response is a 200 OK status with a JSON array of two contact objects. The first contact has `user_id: "venket@gmail.com", name: "harinimathi", email: "harini@gmail.com", phone_no: "7878934894"`. The second contact has `user_id: "venket@gmail.com", name: "sangee", email: "sangeetha@gmail.com", phone_no: "7878934890"`.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> email	venket@gmail.com	
Key	Value	Description

```
1 {
2   {
3     "user_id": "venket@gmail.com",
4     "name": "harinimathi",
5     "email": "harini@gmail.com",
6     "phone_no": "7878934894"
7   },
8   {
9     "user_id": "venket@gmail.com",
10    "name": "sangee",
11    "email": "sangeetha@gmail.com",
12    "phone_no": "7878934890"
13  }
14 }
```

## 5. Update a contact's phone no./name

(Update op.)

i/p: user id, contact's email id, new phone no./name

o/p: Successful updation

The screenshot shows a REST client interface with the URL `http://127.0.0.1:8010/add_contact`. The request method is **PUT**. The request body is a JSON object: `{ "user_id": "venket@gmail.com", "name": "harinimathi", "email": "harini@gmail.com", "phno": "7878934894" }`. The response status is **200 OK** with a response body of `{ "status": "update" }`.

```
1 {
2   "user_id": "venket@gmail.com",
3   "name": "harinimathi",
4   "email": "harini@gmail.com",
5   "phno": "7878934894"
6 }
```

```
1 {
2   "status": "update"
3 }
```

## 6.Delete contact

(Delete op.)

i/p: user id, phone no.

o/p: Successfully deleted contact

The screenshot shows a REST client interface with the URL `http://127.0.0.1:8010/add_contact`. The request method is **DELETE**. The request body is a JSON object: `{ "user_id": "venket@gmail.com", "phno": "7878934890" }`. The response status is **200 OK** with a response body of `{ "status": "delete" }`.

```
1 {
2   "user_id": "venket@gmail.com",
3   "phno": "7878934890"
4 }
5
```

```
1 {
2   "status": "delete"
3 }
```

## 7. Search contact based on email id and name

i/p: user id, keyword

o/p: list of contacts having keyword in name or email.

The screenshot shows a REST client interface with the following details:

- URL:** `http://127.0.0.1:8010/search?id=venket@gmail.com&word=gee`
- Method:** GET
- Query Params:**

KEY	VALUE	DESCRIPTION
id	venket@gmail.com	
word	gee	
- Body:** Pretty view showing a JSON response:

```
1 {
2   "user_id": "venket@gmail.com",
3   "name": "sangee",
4   "email": "sangeetha@gmail.com",
5   "phone_no": 7878934890
6 }
```
- Status:** 200 OK, Time: 515 ms, Size: 336 B

## 8. Check for authentication

i/p: email

o/p: Registered or Not Registered

The screenshot shows a REST client interface with the following details:

- URL:** `http://127.0.0.1:8010/auth`
- Method:** POST
- Body:** Raw view showing a JSON request:

```
1 {
2   "email": "loganaaki@gmail.com"
3 }
```
- Body:** Pretty view showing a JSON response:

```
1 {
2   "status": "not registered"
3 }
```
- Status:** 200 OK, Time: 16 ms, Size: 262 B