```
@app.route('/gconnect', methods=['POST'])
def gconnect():
    # Validate state token
    if request.args.get('state') != login_session['state']:
        response = make_response(json.dumps('Invalid state
parameter.'), 401)
        response.headers['Content-Type'] = 'application/json'
        return response
    # Obtain authorization code
    code = request.data

    try:
        # Upgrade the authorization code into a credentials object
        oauth_flow = flow_from_clientsecrets('client_secrets.json',
scope='')
        oauth_flow.redirect_uri = 'postmessage'
        credentials = oauth_flow.step2_exchange(code)
    except FlowExchangeError:
        response = make_response(
            json.dumps('Failed to upgrade the authorization code.'),
401)
        response.headers['Content-Type'] = 'application/json'
        return response

    # Check that the access token is valid.
    access_token = credentials.access_token
    url = ('https://www.googleapis.com/oauth2/v1/tokeninfo?
access_token=%s'
           % access_token)
    h = httplib2.Http()
    result = json.loads(h.request(url, 'GET')[1])
    print('error in g connect is', result.get('error'))
    # If there was an error in the access token info, abort.
    if result.get('error') is not None:
        response = make_response(json.dumps(result.get('error')), 500)
        response.headers['Content-Type'] = 'application/json'
        print('in result.get("error")')
        return response

    # Verify that the access token is used for the intended user.
    gplus_id = credentials.id_token['sub']
    if result['user_id'] != gplus_id:
        response = make_response(
            json.dumps("Token's user ID doesn't match given user
ID."), 401)
        response.headers['Content-Type'] = 'application/json'
        return response

    # Verify that the access token is valid for this app.
    if result['issued_to'] != CLIENT_ID:
        response = make_response(
            json.dumps("Token's client ID does not match app's."),
401)
        print "Token's client ID does not match app's."
        response.headers['Content-Type'] = 'application/json'
```

```
        return response

    stored_access_token = login_session.get('access_token')
    stored_gplus_id = login_session.get('gplus_id')
    if stored_access_token is not None and gplus_id ==
stored_gplus_id:
        response = make_response(json.dumps
                                 ('Current user is already
connected.'),
                                 200)
        response.headers['Content-Type'] = 'application/json'
        return response

    # Store the access token in the session for later use.
    login_session['access_token'] = access_token
    login_session['gplus_id'] = gplus_id

    # Get user info
    userinfo_url = "https://www.googleapis.com/oauth2/v1/userinfo"
    params = {'access_token': access_token, 'alt': 'json'}
    answer = requests.get(userinfo_url, params=params)

    data = answer.json()

    login_session['username'] = data['name']
    login_session['picture'] = data['picture']
    login_session['email'] = data['email']

    # ADD PROVIDER TO LOGIN SESSION
    login_session['provider'] = 'google'

    # see if user exists, if it doesn't make a new one
    user_id = getUserId(login_session['email'])
    if not user_id:
        user_id = createUser(login_session)
    login_session['user_id'] = user_id

    output = ''
    output += '<h1>Welcome, '
    output += login_session['username']
    output += '!</h1>'
    output += '<img src="'
    output += login_session['picture']
    output += ' " style = "width: 300px; height: 300px;"> '
    flash("you are now logged in as %s" % login_session['username'])
    return output


@app.route('/gdisconnect')
def gdisconnect():
    access_token = login_session.get('access_token')
    if access_token is None:
        print 'Access Token is None'
        response = make_response(
            json.dumps('Current user not connected.'), 401)
        response.headers['Content-Type'] = 'application/json'
        return response
```

```
    print 'In gdisconnect access token is %s', access_token
    print 'User name is: '
    print login_session['username']
    b = login_session['access_token']
    url = 'https://accounts.google.com/o/oauth2/revoke?token=%s' % b
    h = httplib2.Http()
    result = h.request(url, 'GET')[0]
    print result
    if result['status'] == '200':
        del login_session['access_token']
        del login_session['gplus_id']
        del login_session['username']
        del login_session['email']
        del login_session['picture']
        response = make_response(json.dumps('Successfully
disconnected.'), 200)
        response.headers['Content-Type'] = 'application/json'
        return response
    else:
        response = make_response(
            json.dumps(
                'Failed to revoke token for given user.',
                400))
        response.headers['Content-Type'] = 'application/json'
        return response
```