# SSE Sign-off Task: API with Auth

As part of your coursework, we will issue several sign-off tasks. Your solutions to three of these must be signed off. The exercises from lab 8 and previous labs should be helpful with this task.

## Task: Create a service (API) with Authorization

Create a service (API) in Google App Engine that provides random numbers, but only to authorized users; and it allows you (the admin) to manage authorized users. Any user with a Google account can request to be authorized (equivalent to registering).

The implementation need not use any persistent storage, just keep your user list in memory.

### API Operations

| Method | Path | Required Role | Description |
|---|---|---|---|
| `GET` | `/api/random` | `user` | retrieves a random number |
| `GET` | `/api/user/roles` | - | retrieves roles of logged-in user |
| `POST` | `/api/user/request` | - | requests approval for logged-in user (no body) (this will be ignored if the user is already approved – i.e. has some role) |
| `GET` | `/api/users` | `admin` | lists all known users |
| `GET` | `/api/user/request` | `admin` | lists approval requests |
| `POST` | `/api/user/approve` | `admin` | adds a user (email in the body) |
| `DELETE` | `/api/user/user1@a.b` | `admin` | deletes a user (email in the url) |

### Error Status Codes

- `401 unauthorized` returned by any of the above request if the user isn't signed in or if they provide an invalid ID token
- `403 forbidden` returned by any route that requires some role if the user does not have the role.

### Testing

Download and unzip this testing page: it is a test client page that checks that the API works as expected. Your app could serve that webpage as the default page. You need to put your own client ID instead of the string "client-id" on line 4 of index.html.

The page has some automated tests, and a simple UI to test the API manually.

## Example Run-Through

The API needs to maintain a list of approved users and their roles. Users should be identified by email addresses. The server needs to have a pre-defined admin user (probably you) which should initially have both the roles: '`admin`' and '`user`'.

```
// nobody's logged in yet
GET    /api/random          -> 401 unauthorized
GET    /api/user/roles      -> 401 unauthorized
GET    /api/user/request    -> 401 unauthorized
POST   /api/user/request    -> 401 unauthorized

// log in as you (admin)
GET    /api/random          -> 0.90434364  - you have access
GET    /api/user/roles      -> ['admin','user']  - order does not matter
GET    /api/user/request    -> [] - nobody requested approval yet
GET    /api/users           -> [ { "email": "admin@a.b",
                                   "roles": ['user', 'admin'] } ] - the admin's there

// log in as user1 (with some email address user1@a.b different from the admin's)
GET    /api/random          -> 403 forbidden  - you haven't been approved yet
GET    /api/user/roles      -> []
POST   /api/user/request    -> 202 accepted
GET    /api/user/request    -> 403 forbidden  - only admin can do this
GET    /api/users           -> 403 forbidden  - only admin can do this

// log in as you (admin)
GET    /api/users           -> [ { "email": "admin@a.b",
                                   "roles": ['user', 'admin'] },
                                 { "email": "user1@a.b",
                                   "roles": [] } ] - the new user's there
GET    /api/user/request    -> ["user1@a.b"]  - this user has requested authorization
POST   /api/user/approve user1@a.b -> { "email": "user1@a.b",
                                         "roles": ['user'] }  - basically a confirmation
GET    /api/user/request    -> []  - the user was removed when approved
GET    /api/users           -> [ { "email": "admin@a.b",
                                   "roles": ['user', 'admin'] },
                                 { "email": "user1@a.b",
                                   "roles": ['user'] } ] - the new user's there, approved

// log in as user1
GET    /api/user/roles      -> ['user']
GET    /api/random          -> 0.4398629
POST   /api/user/approve ZZZ@a.b -> 403 forbidden  - only admin can do this

// log in as you (admin)
DELETE /api/user/user1@a.b -> 204 no content
GET    /api/users           -> [ { "email": "admin@a.b",
                                   "roles": ['user', 'admin'] } ] - the new user's gone

// log in as user1
GET    /api/user/roles      -> []   - you've been deleted
GET    /api/random          -> 403 forbidden

// log out
GET    /api/random          -> 401 unauthorized
GET    /api/users           -> 401 unauthorized
```