**PomeloTech**

# Overview

*Commandprompt: node main.js*

*url: localhost:3000/#/*

*"Admin@11.com", "password"*

*Zin Tun: first of all I need to edit connection.json*

*Zin Tun: then reset nodejs then run createAdmin.sql and init.sql*

PomeloTech is a new online marketplace for computer technicians to provide their services. The electronically proficient can offer their expertise with those who are in seek of repair services. Pom­eloTech makes this interaction easy and enjoyable.

# Resources for User Management, including Reg­istration

## User

* Collection of all current technicians or consumers.

GET email=<email or email prefix>

* Return a single user whose email matches the provided one as a resource URL. Authorization is only granted to users with admin status.
* Not Found error will be returned should there be no user with the specified email.
* *email* principal string identifier, unique across all Users
* id - id of user with said email, so that URI would be User/<id>

### POST

* Purpose
  + Adds a new User, returning the location of the newly added User.
* No AU required.
* Attributes required for signup
  + *UserId –* unique identifying number for each user
  + *email*- unique Email for new User
  + *firstName-* first name of user
  + *lastName-* last name of user
  + *password-* password
  + *role*0 for consumer, 1 for technician, 2 for admin
  + *WhenRegistered-* When was the user first registered
* Check conditions
  + Email and lastName required. Error if email is non-unique. Missing fields allowed only if AU is admin.

## User/<usrId>

### GET

* + Returns object for User <usrId>, with all fields*.*
  + AU must be User in question or admin.
  + NotFound return if usrId does not exist

### PUT

* + Update User <usrId>, with body giving an object with one or more of the specified fields*.*
  + AU must be the User in question, or an admin.   
    Note: Role changes require an admin.
  + Errors if there is old password is not provide when changing the password

### DELETE

* + Delete the User in question, including all Serve ordered by User.
  + Requires admin AU/User in question.

## User/<usrId>/Serv

All the services that is tied to the usrId.

### GET servName=<name of service>

* + Returns an array of Serv for the specified User.
  + Accept a query params(status) to get all the working services.
  + AU must be User in question or admin.

### POST

* + Creates a new Serv for this User, on the specified User. Fail if the person in question has more than 5 services. Services must have unique names.

## Ssns

Login sessions (Ssns)are required for most serviceaccess. A user obtains one via POST to Ssns.

### GET

* + Returns a list of all active sessions. Admin-privileged AU required. Returns array of
  + *cookie*Unique cookie value for session
  + *usrId* ID of User logged in

### POST

* A successful POSTgenerates a browser-session cookie that will permit continued access for 2hours. Indicated User becomes the AU. An unsuccessful POSTresults ina400/No Permissionerror code, with no further information.
* *email* Email of user requesting login
* *password* Password of user

## Ssns/<cookie>

### DELETE

* Log out the specified Ssn. AU must be owner of Ssn or admin.

# Resources for Services

## Serv/

### GET (services table)

* Retrieve all the Services in the database. AU must be admin.

**Serv/<techId>**

### GET (servicesOffer table)

* Retrieve all the services offer by techId. User must log in.

**Serv/<userId>**

### GET (servicesOffer table)

* Retrieve all the services requested by userId. User must log in.

**Serv/<servId>/Services**

### GET (servicesOffer table)

* Retrieve all the Services in the database associated to servId. User must log in.

## Serv/<ServId>

### GET

* Retrieve Serv-specific info. AU must be Serv owner or admin. Returns
  + *Services*URI of the challenge being played,
  + *userId*ID of the User making the Serv
  + *status*0 for open, 1 pending, 2 closed
  + *timeStamp*Time of Serv start
  + *technicianId of the creator of a service*

### DELETE

* Delete a service in Services table specified by <Servld>. AU must be the technician who created that service or admin.

**Serv/<servId>/Order**

### PUT

* Customer place the order. ServiceId status in servicesOffer changed.

**DELETE**.

* + Can delete the services associated with servId. User must login.
  + If logged in user.role == 1 or 2, delete the whole row
  + If logged in user.role== 0, update the status from pending to open

# Special DB Resource for Testing Purposes

**DB**

## DELETE

Clear all content from the database, reset all auto increment IDs to 1, and add back one User, an admin named Admin IAM with email admin@11.com and password "password". Clear all current sessions. AU must be an admin.