Our Web Services API

The path to the resources:

Nigel's server: eeyore.fost.plymouth.ac.uk:8082/
On your own machine: localhost:8080/

Then the following will always be the same added on to one of the above:

PRCSA_API/resources/

The complete URL will be ass follows…

eeyore.fost.plymouth.ac.uk:8082/PRCSA_API/resources/{Here will be the specific resource}

All web services are lower case and case sensitive.
Wherever you see { id } then the braces are only to indicate you should put your own id in there. Remove the braces from your URL.

Gets:

1. allrules
2. currentadvertsimages
3. currentadvertsimages/{member_id}
4. pastadvertsimages
5. pastadvertsimages/{member_id}
6. currentadverts
7. currentadverts/{member_id}
8. pastadverts
9. pastadverts/{ member_id}
10. memberpassword/{email
11. memberdetails/id_{member_id}
12. memberdetails/{email}
13. recordsession/{member_id}
14. searchadverts/{tags}
15. searchadvertsimages/{tags}
16. membertransactions/incoming/{id}
17. membertransactions/outgoing/{id}

Posts:

1. createadvert
2. registermember
3. updatemember/name
4. updatemember/email
5. updatemember/password
6. updatemember/address
7. updatemember/telephone
8. uploadadvertimage

If you use my data model then you can use Gson to automatically convert the result of these queries into the objects they represent.

Example: Rules rule = new Rules(); ||
Convert rule to JSON using Gson. || **These 3 are already done for you.**
Use web service to transfer the data. ||
Use Gson to convert JSON string into Rules object.

Gets:

allrules – Gets all the rules in the database


```
{
	id: 1
	rule: "OIfoihfiohs[oishfaoiha["
}
{
	id: 109
	rule: "Testing to see if a change is being made."
}
```


currentadvertsimages – Gets all the current adverts images in the database.

currentadvertsimages/{member_id} – Gets this members current adverts images.

pastadvertsimages – Gets all the past adverts images in the database.

pastadvertsimages/{member_id} – Gets this members past adverts images.

All of the above requests return the following except the ones with an ID will only return one set of details. The details are the adverts id and the image encoded as a Bas64Binary.


```
"24"
"/9j/4AAQSkZJRgABAQEAYABgAAD/4QAiRXhpZgAATU0AKgAAAAgAAQESAAMAAAABAAEAAAAAAAD/7AARRHVja3
kAAQAEAAAAOgA. . . . etc

"61"
"/9j/4AAQSkZJRgABAQEASABIAAD/4QnMRXhpZgAATU0AKgAAAAgACAEOAAIAAAAiAAAAbgESAAMAAAABAAEAAA
EaAAUAAAABAAAAkAEbA . . . etc
```


currentadverts – Gets the details of all current adverts. Getting all will return as shown in the image below.

currentadverts/{member_id} – Gets all the details of this members current adverts.

pastadverts – Gets the details of all past adverts. Getting all will return as shown in the image below.

pastadverts/{ member_id} – Gets all the details of this members past adverts.

```
{
	id: 24
	title: "Accountings"
	description: "Provide accounting advice"
	member_id: 5
	date_adv: "Feb 2, 2015 12:00:00 AM"
	advert_type: "Offer"
	category: "Computing And Electronics"
	item_type: "Service"
	date_exp: "Mar 2, 2015 12:00:00 AM"
	cost: 1
	transport: false
	advert_tags: "Accounting,Advice"
	is_active: "Y"
}
{
	id: 61
	title: "House Removals"
	. . . more details
}
{
	id: 43
	title: "Walking Dogs"
	. . . more details
}
```


memberpassword/{email} – Gets a members password using their email. Only email is possible as the user table is not linked to members and only the email is stored with the password.


```
"$2a$12$J9ljqEIqT8TMEaBnj0ETJuUMmejPvesyiVL4wfcyeTgPdLin.6e4C"
```


memberdetails/id_{member_id} – Gets a members details using their ID. Take note of the id_ before your entered ID.

memberdetails/{email} – Gets a members details using their email address.


```
{
	addline1: "30 Clifton Place"
	addline2: "North Hill"
	balance: 10
	city: "Plymouth"
	contact_number: "07767912912"
	dob: "1989-05-31"
	email: "brianviv@gmail.com"
	forename: "Brians"
	id: 6
```

```
        is_active: "Y"
        postcode: "PL2 7JH"
        rating: 0
        surname: "Viviers"
}
```


recordsession/{member_id} – When the user logs in call this web service with the members ID to record that they have logged into the system. No return is given.

searchadverts/{tags} - Search for adverts using a search term

tags could be: just one word such as “accounting” OR multiple words and phrases separated by commas such as: “van hire, java programming, accounting”
NB You must URLEncode.encode the search tags as they will contain white spaces.

searchadvertsimages/{tags} – Gets the images related to the searched for adverts.

membertransactions/incoming/{id} – Gets transactions that the member will receive money from

membertransactions/outgoing/{id} - Gets transactions that the member will have to pay money.

Posts:

registermember – Used to register a member
The user_access table will need to be updated at the same time with the email and hashed password.

createadvert – Used to create a new advert
Required: title, description, category, advert_type, item_type, credits, member_id, transport.
You will need to ask me about this one.

updatemember/name – Post a new forename and surname in a member object
Required: forename & surname & member_id

updatemember/email - Post a new email in a member object
Required: new email & member_id

updatemember/password - Post a new password in a member object
Required: hashed password and email

updatemember/address - Post a address in a member object
Required: All four lines of the address & member_id

updatemember/telephone - Post a new telephone number in a member object
Required: All new number & member_id

Uploadadvertimage - Post a new image to an existing advert.