Unified Cognitive Assessment Research Platform API Specification

Repository URL: https://bitbucket.org/guana/phydsl-games
API file: ucap-backend/ucap/api.py

- API Summary
- Methods
- Test Cases
  - Bad Requests
  - Successful Requests
- cURL Examples

API Summary

Note: For this document, session cookies are not considered as input to the methods that take them, as the browser or tablet passes them automatically if present.

Endpoint	Description
/api/login	Log in to UCAP (website or tablet game). Takes the tester's username and password as input, and creates a session cookie with a corresponding session in the database. Does not require authentication.
/api/logout	Log out of UCAP (website or tablet game). Takes no input, and destroys the database session matching the supplied cookie. Does not require authentication (although it will have no effect if not authenticated).
/api/check_session	Check if logged in. Takes no input, and indicates whether or not the tester is logged in based on the presence or absence of a session cookie. Does not require authentication.
/api/create_completed_test	Persist a test result. Takes a unique game identifier, the tested patient ID, the test date, and any game-specific parameters, and persists the test result. Requires authentication.
/api/get_tests	Get a collection of test results for a particular game. Takes a unique game identifier and a tester- or patient-related string to filter the test results by (optional), and returns a collection of test results. Requires authentication.
/api/create_patient	Create a patient. Takes a tester-chosen ID for the patient, an indicator as to whether the

	patient will be tested with blind tests or non-blind tests, the patient’s name (optional), group (optional), gender (optional), date of birth (optional), and notes about the patient (optional). Persists the patient and generates a patient-specific password (only for non-blind patients). Other testers must unlock the patient with this password to test them. Requires authentication.
/api/get_patient/:id	Get a single patient. Takes the ID of the desired patient and returns the patient. Requires authentication.
/api/get_patients	Get a collection of patients. Takes an indicator as to whether the patients are being displayed as part the test setup process (if so, blind patients the logged-in tester created won’t be displayed) and the patient type (non-blind or blind) to filter for (optional). Returns a collection of patients. Requires authentication.
/api/unlock_patient/:id	Unlock a patient. Takes the ID of the patient to unlock and the patient’s password. Makes the patient’s information visible to the tester and lets the tester test the patient. Requires authentication.
/api/search_patients	Get a filtered collection of patients. Takes a tester-related string to filter the patients by and returns the collection. Requires authentication.
/api/create_tester	Create a tester. Takes the tester’s email, organization, first and last name, chosen username, and chosen password (repeated), and persists the tester. Does not require authentication.

/api/get_tester/:id	Get a single tester. Takes the ID of the desired tester and returns the tester. Requires authentication.
/api/get_testers	Get a collection of testers. Takes no input and returns the collection. Requires authentication.
/api/search_testers	Get a filtered collection of testers. Takes a patient-related string to filter the testers by and returns the collection. Requires authentication.

Methods

Note: Unless otherwise stated, all parameters are required.

Method 1 - Log in to UCAP

Endpoint: /api/login

Parameters:

- username: The tester's username
- password: The tester's password

Output: Creates a session cookie with a corresponding session in the database

Requires authentication? No

Request Example:

```
Request:

POST /api/login

{
  "username": "freud"
  "password": "asdfasdf"
}
```

Success Response Example:

```
Response:

HTTP/1.1 200 OK
```

Failure Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
Content-Type: application/json
{
        "errors": {"username_or_password": "Invalid username or password"}
}
```

Method 2 - Log out of UCAP

Endpoint: /api/logout

Parameters: None

Output: Destroys the database session matching the supplied cookie (if the cookie exists)

Requires authentication? No

Request Example:

```
Request:

POST /api/logout
```

Success Response Example:

```
Response:

HTTP/1.1 200 OK
```

Method 3 - Check if logged in

Endpoint: /api/check_session

Parameters: None

Output: Indicates whether or not the tester is logged in based on the presence or absence of a session cookie

Requires authentication? No

Request Example:

```
Request:

POST /api/check_session
```

Success Response Example:

```
Response:

HTTP/1.1 200 OK

Content-Type: application/json
{
        "is_logged_in": true
}
```

<u>Method 4 - Persist a test result</u>

Endpoint: /api/create_completed_test

Parameters:

- app: The unique identifier for the game the test result is from (e.g. “star” for Star Cancellation)
- test_date: The date of the test administration, as a YYYY-MM-DD string
- patient_id: The id of the patient who took the test

Other, game-specific parameters must be supplied. For example, Star Cancellation requires:

- elapsed_time: How long the patient took to do the test, in seconds
- score_total: The number of stars the patient cancelled
- score_expected: The number of stars available to cancel
- score_zones: The patient’s score by left/right zone
- perseverations: The number of times the patient cancelled the same star more than once.
- latency_average: The average time between cancellations, in seconds
- latency_sd: The standard deviation of the the time between cancellations, in seconds
- events: A list of the cancellation events

Output: Persists the test result

Requires authentication? Yes

Request Example:

```
Request:

POST /api/create_completed_test

{
  "app": "star",
  "patient_id": 1,
  "test_date": "2016-05-01",
  "elapsed_time": 501,
  "score_total": 4,
  "score_expected": 51,
  "score_zones": "L:0/0/3 - R:1/0/0",
  "perseverations": 0,
  "latency_average": 0.223,
  "latency_sd": 0.101,
  "events": "[8.1:5.5:1461250977726, 5.4:4.9:1461250978020, 11.1:4.6:1461250978353,
6.2:5.9:1461250978442, 6.1:3.7:1461250978688, 4.8:2.5:1461250978839]"
}
```

Success Response Example:

```
Response:

HTTP/1.1 201 Created

Content-Type: application/json

{
  "test_id": 1
}
```

Failure Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 5 - Get a collection of test results for a particular game

Endpoint: /api/get_tests

Parameters:

- app_code: The unique identifier for the game the test result is from (e.g. “star” for Star Cancellation)
- query: A tester- or patient-related string to filter the test results by (optional)

Output: A collection of test results

Requires authentication? Yes

Request Example:

```
Request:

GET /api/get_tests?app_code=star
```

Success Response Example:

```
Response:

HTTP/1.1 200 OK

Content-Type: application/json

{"Tests": [{"id": 1, "app": "star", "patient_id": 1, "tester_id": 1, ...}, ...]}
```

Failure Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 6 - Create a patient

Endpoint: /api/create_patient

Parameters:

- visible_patient_id: The tester-chosen ID for the patient
- patient_type: Whether the patient will be tested with blind tests (blind) or non-blind tests (non-blind)
- name: The patient's name (optional)
- group: The patient’s group (optional)
- gender: The patient's gender (optional)
- dob: The patient's date of birth (optional)
- notes: Notes about the patient (optional)

Output: Persists the patient and generates a patient-specific password (only for non-blind patients). Other testers must unlock the patient with this password to test them.

Requires authentication? Yes

Request Example:

```
Request:

POST /api/create_patient

{
  "visible_patient_id": "A1",
  "patient_type": "non-blind",
  "name": "Jane Doe",
  "group": "Control1"
  "gender": "female",
  "dob": "1930-01-01"
}
```

Success Response Example:

```
Response:

HTTP/1.1 201 Created

Content-Type: application/json

{
  "id": 1,
  "password": "asdfqwer"
}
```

Failure Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

<u>Method 7 - Get a single patient</u>

Endpoint: /api/get_patient/:id

Parameters:

- patient_id: The ID of the desired patient

Output: The desired patient

Requires authentication? Yes

Request Example:

```
Request:

GET /api/get_patient?patient_id=1
```

Success Response Example:

```
Response:
```

```
HTTP/1.1 200 OK

Content-Type: application/json

{
  "Patient":
  {
    "id": 1,
    "patient_type": "non-blind",
    "visible_patient_id": "A1",
    "name": "Jane Doe",
    "group": "Control1",
    "gender": "female",
    "dob": "1930-01-01"
  }
}
```

Failure Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

<u>Method 8 - Get a collection of patients</u>

Endpoint: /api/get_patients

Parameters:

- test_selection: Whether the patients are being displayed as part the test setup process; if so, blind patients the logged-in tester created won’t be displayed
- patient_type: The type of patient (non-blind or blind) to filter for (optional)

Output: A collection of patients

Requires authentication? Yes

Request Example:

```
Request:

GET /api/get_patients?test_selection=0
```

Success Response Example:

```
Response:
```

```
HTTP/1.1 200 OK

Content-Type: application/json

{"Patients": [{"id": 1, "patient_type": "non-blind", "visible_patient_id": "A1", ...}, ...]}
```

Failure Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 9 - Unlock a patient

Endpoint: /api/unlock_patient:id

Parameters:

- patient_id: The ID of the patient to unlock
- password: The patient's password

Output: Makes the patient's information visible to the tester and lets the tester test the patient

Requires authentication? Yes

Request Example:

```
Request:

POST /api/unlock_patient

{
  "patient_id": 1,
  "password": "asdfqwer"
}
```

Success Response Example:

```
Response:

HTTP/1.1 201 Created

Content-Type: application/json

{
  "unlock_sucessful": true
}
```

Failure Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

<u>Method 10 - Get a filtered collection of patients</u>

Endpoint: /api/search_patients

Parameters:

- query: A tester-related string to filter the patients by

Output: A collection of patients

Requires authentication? Yes

Request Example:

```
Request:

GET /api/search_patients?query=freud
```

Success Response Example:

```
Response:

HTTP/1.1 200 OK

Content-Type: application/json

{"Patients": [{"id": 1, "patient_type": "non-blind", "visible_patient_id": "A1", ...}, ...]}
```

Failure Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

<u>Method 11 - Create a tester</u>

Endpoint: /api/create_tester

Parameters:

- email: The tester’s email

- organization: The tester’s organization
- first_name: The tester’s first name
- last_name: The tester’s last name
- username: The tester’s chosen username
- password: The tester’s chosen password
- password_again: The same password as above

Output: Persists the tester.

Requires authentication? No

Request Example:

```
Request:

POST /api/create_tester


{
  "email": "freud@example.com",
  "organization": "U of A",
  "first_name": "Sigmund",
  "last_name": "Freud",
  "username": "freud",
  "password": "asdfasdf",
  "password_again": "asdfasdf"
}
```

Success Response Example:

```
Response:

HTTP/1.1 201 Created

Content-Type: application/json

{
  "tester_id": 1
}
```

Failure Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
```

```
{
        "errors': {'missing_fields': 'Please supply all required fields'}
}
```

Method 12 - Get a single tester

Endpoint: /api/get_tester/:id

Parameters:

- tester_id: The ID of the desired tester

Output: The desired tester

Requires authentication? Yes

Request Example:

```
Request:

GET /api/get_tester?tester_id=1
```

Success Response Example:

```
Response:

HTTP/1.1 200 OK

Content-Type: application/json

{
  "Tester":
  {
        "organization": "U of A",
        "username": "freud",
        "first_name": "Sigmund",
        "last_name": "Freud"
  }
}
```

Failure Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 13 - Get a collection of testers

Endpoint: /api/get_testers

Parameters: None

Output: A collection of testers

Requires authentication? Yes

Request Example:

```
Request:

GET /api/get_testers
```

Success Response Example:

```
Response:

HTTP/1.1 200 OK

Content-Type: application/json

{"Testers": [{"organization": "U of A", "username": "freud", "first_name": "Sigmund", "last_name":
Freud"}, ...]}
```

Failure Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

<u>Method 14 - Get a filtered collection of testers</u>

Endpoint: /api/search_testers

Parameters:

- query: A patient-related string to filter the testers by

Output: A collection of testers

Requires authentication? Yes

Request Example:

```
Request:

GET /api/search_testers?query=jane
```

Success Response Example:

```
Response:
```

```
HTTP/1.1 200 OK

Content-Type: application/json

{"Testers": [{"organization": "U of A", "username": "freud", "first_name": "Sigmund", "last_name":
Freud"}, ...]}
```

Failure Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Test Cases

Bad Requests

Method 1: Log in to UCAP

Error 1: Invalid username

Request Example:

```
Request:

POST /api/login


{
  "username": "wrong_username"
  "password": "asdfasdf"
}
```

Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
Content-Type: application/json
{
        “errors”: {“username_or_password”: “Invalid username or password”}
}
```

Method 4: Persist a test result

Error 1: Not logged in

Request Example:

```
Request:

POST /api/create_completed_test


Content-Type: application/json


{
  "app": "star",
  "patient_id": 1,
  "test_date": "2016-05-01",
  "elapsed_time": 501,
  "score_total": 4,
  "score_expected": 51,
  "score_zones": "L:0/0/3 - R:1/0/0",
  "perseverations": 0,
  "latency_average": 0.223,
  "latency_sd": 0.101,
  "events": "[8.1:5.5:1461250977726, 5.4:4.9:1461250978020, 11.1:4.6:1461250978353,
6.2:5.9:1461250978442, 6.1:3.7:1461250978688, 4.8:2.5:1461250978839]"
}
```

Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 4: Persist a test result

Error 2: Missing field(s)

Request Example:

```
# No patient ID

Request:

POST /api/create_completed_test

Content-Type: application/json

{
  "app": "star",
  "test_date": "2016-05-01",
  "elapsed_time": 501,
  "score_total": 4,
  "score_expected": 51,
  "score_zones": "L:0/0/3 - R:1/0/0",
  "perseverations": 0,
  "latency_average": 0.223,
  "latency_sd": 0.101,
  "events": "[8.1:5.5:1461250977726, 5.4:4.9:1461250978020, 11.1:4.6:1461250978353,
6.2:5.9:1461250978442, 6.1:3.7:1461250978688, 4.8:2.5:1461250978839]"
}
```

Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
{
        'errors': {'missing_fields': 'Please supply all required fields'}
}
```

Method 5: Get a collection of test results for a particular game

Error 1: Not logged in

Request Example:

```
Request:

GET /api/get_tests?app_code=star
```

Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 5: Get a collection of test results for a particular game

Error 2: Missing field(s)

Request Example:

```
# No app code

Request:

GET /api/get_tests
```

Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
{
        'errors': {'missing_fields': 'Please supply all required fields'}
}
```

Method 6: Create a patient

Error 1: Not logged in

Request Example:

```
Request:

POST /api/create_patient

{
  "visible_patient_id": "A1",
  "patient_type": "non-blind",
  "name": "Jane Doe",
  "group": "Control1"
  "gender": "female",
  "dob": "1930-01-01"
}
```

Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 6: Create a patient

Error 2: Missing field(s)

Request Example:

```
# No visible patient ID

Request:

POST /api/create_patient
```

```
{
  "patient_type": "non-blind",
  "name": "Jane Doe",
  "group": "Control1"
  "gender": "female",
  "dob": "1930-01-01"
}
```

Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
{
        'errors': {'missing_fields': 'Please supply all required fields'}
}
```

Method 6: Create a patient

Error 3: Invalid field(s)

Request Example:

```
# Invalid gender and DOB

Request:

POST /api/create_patient

{
  "patient_type": "non-blind",
  "name": "Jane Doe",
  "group": "Control1"
  "gender": "dragonkin",
  "dob": "A long time ago"
}
```

Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
```

```
{
        'errors': {'gender': 'Invalid gender', 'dob': 'Invalid DOB'}
}
```

Method 7: Get a single patient

Error 1: Not logged in

Request Example:

```
Request:

GET /api/get_patient?patient_id=1
```

Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 7: Get a single patient

Error 2: Missing field(s)

Request Example:

```
# No patient ID

Request:

GET /api/get_patient
```

Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
{
        'errors': {'missing_fields': 'Please supply all required fields'}
}
```

Method 8: Get a collection of patients

Error 1: Not logged in

Request Example:

```
Request:

GET /api/get_patients?test_selection=0
```

Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 8: Get a collection of patients

Error 2: Missing field(s)

Request Example:

```
# No test_selection

Request:

GET /api/get_patients
```

Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
{
        'errors': {'missing_fields': 'Please supply all required fields'}
}
```

Method 9: Unlock a patient

Error 1: Not logged in

Request Example:

```
Request:

POST /api/unlock_patient
```

```
{
  "patient_id": 1,
  "password": "asdfqwer"
}
```

Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 9: Unlock a patient

Error 2: Missing field(s)

Request Example:

```
# No password

Request:

POST /api/unlock_patient

Content-Type: application/json

{
  "patient_id": 1
}
```

Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
{
        'errors': {'missing_fields': 'Please supply all required fields'}
}
```

Method 9: Unlock a patient

Error 3: Wrong password

Request Example:

```
Request:

POST /api/unlock_patient


{
  "patient_id": 1,
  "password": "wrong_password"
}
```

Response Example:

```
Response:

HTTP/1.1 200 OK


Content-Type: application/json
{
        "unlock_successful": false
}
```

Method 10: Get a filtered collection of patients

Error 1: Not logged in

Request Example:

```
Request:

GET /api/search_patients?query=freud
```

Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 10: Get a filtered collection of patients

Error 2: Missing field(s)

Request Example:

```
# No query

Request:

GET /api/search_patients
```

Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
{
        'errors': {'missing_fields': 'Please supply all required fields'}
}
```

Method 11: Create a tester

Error 1: Missing field(s)

Request Example:

```
# No username

Request:

POST /api/create_tester

{
  "email": "freud@example.com",
  "organization": "U of A",
  "first_name": "Sigmund",
  "last_name": "Freud",
  "password": "asdfasdf",
  "password_again": "asdfasdf"
}
```

Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
{
        'errors': {'missing_fields': 'Please supply all required fields'}
}
```

Method 11: Create a tester

Error 2: Invalid field(s)

Request Example:

```
# Passwords don't match

Request:

POST /api/create_tester

{
  "email": "freud@example.com",
  "organization": "U of A",
  "first_name": "Sigmund",
  "last_name": "Freud",
  "username": "freud",
  "password": "asdfasdf",
  "password_again": "qwerqwer"
}
```

Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
{
        'errors': {'password': 'Passwords don\'t match'}
}
```

Method 11: Create a tester

Error 3: Email or username taken

Request Example:

```
# Email taken

Request:

POST /api/create_tester

{
  "email": "already_taken@example.com",
  "organization": "U of A",
  "first_name": "Sigmund",
  "last_name": "Freud",
  "username": "freud",
  "password": "asdfasdf",
  "password_again": "asdfasdf"
}
```

Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
{
        'errors': {'email': 'Email already exists'}
}
```

Method 12: Get a single tester

Error 1: Not logged in

Request Example:

```
Request:

GET /api/get_tester?tester_id=1
```

Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 12: Get a single tester

Error 2: Missing field(s)

Request Example:

```
# No patient ID

Request:

GET /api/get_tester
```

Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
{
        'errors': {'missing_fields': 'Please supply all required fields'}
}
```

Method 13: Get a collection of testers

Error 1: Not logged in

Request Example:

```
Request:

GET /api/get_testers
```

Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 14: Get a filtered collection of testers

Error 1: Not logged in

Request Example:

```
Request:

GET /api/search_testers?query=jane
```

Response Example:

```
Response:

HTTP/1.1 401 Unauthorized
```

Method 14: Get a filtered collection of testers

Error 2: Missing field(s)

Request Example:

```
# No query

Request:

GET /api/search_testers
```

Response Example:

```
Response:

HTTP/1.1 400 Bad Request

Content-Type: application/json
{
        'errors': {'missing_fields': 'Please supply all required fields'}
}
```

Successful Requests

See “Methods” for examples of successful requests and their responses.

cURL Examples

Note: Methods that require authentication can be executed by first logging in using the cURL command for Method 1 (after creating a tester using the command for Method 11). This will create a file called `ucap_headers` in your working directory that contains a cookie with a valid session ID. This file must be kept in tact and in the same location when executing the authentication-protected methods.

Method 1: Log into UCAP

Supplied parameters (added with -d):

1. username: The tester's username
2. password: The tester's password

```
curl -i -H "Content-Type: application/json" -X POST -D ucap_headers -d '{"username":
"freud", "password": "asdfasdf"}' http://162.246.156.143/api/login
```

Method 2: Log out of UCAP

Supplied parameters (added with -d): None

```
curl -i -H "Content-Type: application/json" -X POST -b ucap_headers
http://162.246.156.143/api/logout
```

Method 3: Check if logged in

Supplied parameters (added with -d): None

```
curl -i -H "Content-Type: application/json" -X POST -b ucap_headers
http://162.246.156.143/api/check_session
```

Method 4: Persist a test result (Star Cancellation)

Supplied parameters (added with -d):

- app: The unique identifier for the game the test result is from (e.g. "star" for Star Cancellation)
- test_date: The date of the test administration, as a YYYY-MM-DD string
- patient_id: The id of the patient who took the test
- elapsed_time: How long the patient took to do the test, in seconds
- score_total: The number of stars the patient cancelled

- score_expected: The number of stars available to cancel
- score_zones: The patient’s score by left/right zone
- perseverations: The number of times the patient cancelled the same star more than once.
- latency_average: The average time between cancellations, in seconds
- latency_sd: The standard deviation of the the time between cancellations, in seconds
- events: A list of the cancellation events

```
curl -i -H "Content-Type: application/json" -X POST -b ucap_headers -d '{ "app": "star",
"patient_id": 1, "test_date": "2016-05-01", "elapsed_time": 501, "score_total": 4,
"score_expected": 51, "score_zones": "L:0/0/3 - R:1/0/0", "perseverations": 0,
"latency_average": 0.223, "latency_sd": 0.101, "events": "[8.1:5.5:1461250977726,
5.4:4.9:1461250978020, 11.1:4.6:1461250978353, 6.2:5.9:1461250978442, 6.1:3.7:1461250978688,
4.8:2.5:1461250978839]"}' http://162.246.156.143/api/create_completed_test
```

Method 5: Get a collection of test results for a particular game

Supplied parameters (added with -d):

- app_code: The unique identifier for the game the test result is from (e.g. “star” for Star Cancellation)

```
curl -i -b ucap_headers "http://162.246.156.143/api/get_tests?app_code=star"
```

Method 6: Create a patient

Supplied parameters (added with -d):

- visible_patient_id: The tester-chosen ID of the new patient
- patient_type: Whether the patient will be tested with blind tests (blind) or non-blind tests (non-blind)
- name: The patient's name

```
curl -i -H "Content-Type: application/json" -X POST -b ucap_headers -d
'{"visible_patient_id": "A1", "patient_type": "non-blind", "name": "Jane Doe"}'
http://162.246.156.143/api/create_patient
```

Method 7: Get a single patient

Supplied parameters (added with -d):

- patient_id: The ID of the desired patient

```
curl -i -b ucap_headers "http://162.246.156.143/api/get_patient?patient_id=1"
```

Method 8: Get a collection of patients

Supplied parameters (added with -d):

- test_selection: Whether the patients are to be displayed as part the test setup process; if so, blind patients the logged-in tester created won't be displayed

```
curl -i -b ucap_headers "http://162.246.156.143/api/get_patients?test_selection=0"
```

Method 9: Unlock a patient

Supplied parameters (added with -d):

- patient_id: The ID of the patient to unlock
- password: The patient's password

```
curl -i -H "Content-Type: application/json" -X POST -b ucap_headers -d '{"patient_id": 1,
"password":"asdfqwer"}' http://162.246.156.143/api/unlock_patient
```

Method 10: Get a filtered collection of patients

Supplied parameters (added with -d):

- query: A tester-related string to filter the patients by

```
curl -i -b ucap_headers "http://162.246.156.143/api/search_patients?query=freud"
```

Method 11: Create a tester

Supplied parameters (added with -d):

- email: The tester's email
- organization: The tester's organization
- first_name: The tester's first name
- last_name: The tester's last name
- username: The tester's chosen username
- password: The tester's chosen password
- password_again: The same password as above

```
curl -i -H "Content-Type: application/json" -X POST -d '{"email": "freud@example.com",
"organization": "U of A", "first_name": "Sigmund", "last_name": "Freud", "username":
"freud", "password": "asdfasdf", "password_again": "asdfasdf"}'
http://162.246.156.143/api/create_tester
```

Method 12: Get a single tester

Supplied parameters (added with -d):

- tester_id: The ID of the desired tester

```
curl -i -b ucap_headers "http://162.246.156.143/api/get_tester?tester_id=1"
```

Method 13: Get a collection of testers

Supplied parameters (added with -d): None

```
curl -i -b ucap_headers "http://162.246.156.143/api/get_testers"
```

Method 14: Get a filtered collection of testers

Supplied parameters (added with -d):

- query: A patient-related string to filter the testers by

```
curl -i -b ucap_headers "http://162.246.156.143/api/search_testers?query=jane"
```