**Locations**

URIs relative to <http://localhost:8080>

Location Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /location

Request Body:

{ “name” : string}

Will Return 400 Error if location is already in Database

**Get Method**

HTTP Request GET /location

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /location

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if location does not exist in Database

**Languages**

URIs relative to <http://localhost:8080>

Language Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /language

Request Body:

{ “name” : string}

Will Return 400 Error if language is already in Database

**Get Method**

HTTP Request GET /language

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /language

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if language does not exist in Database

**Race**

URIs relative to <http://localhost:8080>

Race Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /race

Request Body:

{ “name” : string}

Will Return 400 Error if race is already in Database

**Get Method**

HTTP Request GET /race

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /race

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if race does not exist in Database

**Ethnicity**

URIs relative to <http://localhost:8080>

Ethnicity Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /ethnicity

Request Body:

{ “name” : string}

Will Return 400 Error if ethnicity is already in Database

**Get Method**

HTTP Request GET /ethnicity

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /ethnicity

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if ethnicity does not exist in Database

**Smoking History**

URIs relative to <http://localhost:8080>

Smoking history Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /smoking-history

Request Body:

{ “name” : string}

Will Return 400 Error if smoking-history is already in Database

**Get Method**

HTTP Request GET /smoking-history

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /smoking-history

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if smoking-history does not exist in Database

**Appetite**

URIs relative to <http://localhost:8080>

Appetite Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /appetite

Request Body:

{ “name” : string}

Will Return 400 Error if appetite is already in Database

**Get Method**

HTTP Request GET /appetite

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /appetite

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if appetite does not exist in Database

**Supplements**

URIs relative to <http://localhost:8080>

Supplement Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /supplement

Request Body:

{ “name” : string}

Will Return 400 Error if supplement is already in Database

**Get Method**

HTTP Request GET /supplement

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /supplement

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if supplement does not exist in Database

**Musculoskeletal**

URIs relative to <http://localhost:8080>

Musculoskeletal Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /musuloskeletal

Request Body:

{ “name” : string}

Will Return 400 Error if musuloskeletal is already in Database

**Get Method**

HTTP Request GET /musuloskeletal

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /musuloskeletal

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if musuloskeletal does not exist in Database

**Bed**

URIs relative to <http://localhost:8080>

Bed Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /bed

Request Body:

{ “name” : string}

Will Return 400 Error if bed is already in Database

**Get Method**

HTTP Request GET /bed

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /bed

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if bed does not exist in Database

**Chair**

URIs relative to <http://localhost:8080>

Chair Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /chair

Request Body:

{ “name” : string}

Will Return 400 Error if chair is already in Database

**Get Method**

HTTP Request GET /chair

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /chair

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if chair does not exist in Database

**Feet**

URIs relative to <http://localhost:8080>

Feet Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /feet

Request Body:

{ “name” : string}

Will Return 400 Error if feet is already in Database

**Get Method**

HTTP Request GET /feet

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /feet

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if feet does not exist in Database

**Past Medical History**

URIs relative to <http://localhost:8080>

Past Medical History Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /past-medical-history

Request Body:

{ “name” : string}

Will Return 400 Error if Past Medical History is already in Database

**Get Method**

HTTP Request GET /past-medical-history

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /past-medical-history

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if Past Medical History does not exist in Database

**Family Medical History**

URIs relative to <http://localhost:8080>

Family Medical History Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /family-medical-history

Request Body:

{ “name” : string}

Will Return 400 Error if Family Medical History is already in Database

**Get Method**

HTTP Request GET /family-medical-history

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /family-medical-history

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if Family Medical History does not exist in Database

**Social History**

URIs relative to <http://localhost:8080>

Social History Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /social-history

Request Body:

{ “name” : string}

Will Return 400 Error if Social History is already in Database

**Get Method**

HTTP Request GET /social-history

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /social-history

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if Social History does not exist in Database

**Gastural Intestinal**

URIs relative to <http://localhost:8080>

Gastural Intestinal Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /gastural-intestinal

Request Body:

{ “name” : string}

Will Return 400 Error if Gastural Intestinal is already in Database

**Get Method**

HTTP Request GET /gastural-intestinal

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /gastural-intestinal

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if Gastural Intestinal does not exist in Database

**Allergies**

URIs relative to <http://localhost:8080>

Allergy Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /allergy

Request Body:

{ “name” : string}

Will Return 400 Error if Allergy is already in Database

**Get Method**

HTTP Request GET /allergy

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /allergy

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if Allergy does not exist in Database

**Anticoagulant**

URIs relative to <http://localhost:8080>

Anticoagulant Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /anticoagulant

Request Body:

{ “name” : string}

Will Return 400 Error if Anticoagulant is already in Database

**Get Method**

HTTP Request GET /anticoagulant

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /anticoagulant

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if Anticoagulant does not exist in Database

**Appearance**

URIs relative to <http://localhost:8080>

Apperaance Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /apperance

Request Body:

{ “name” : string}

Will Return 400 Error if appearance is already in Database

**Get Method**

HTTP Request GET /appearance

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /appearance

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if appearance does not exist in Database

**Vitals**

URIs relative to <http://localhost:8080>

Vitals Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /vital

Request Body:

{ “name” : string}

Will Return 400 Error if Vital is already in Database

**Get Method**

HTTP Request GET /vital

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /vital

Request Body:

{“name” : string}

Response Body:

If successful, empty response body

Will return 400 error if vital does not exist in Database

**Orientation**

URIs relative to <http://localhost:8080>

Orientation Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /orientation

Request Body:

{ “name” : string}

Will Return 400 Error if orientation is already in Database

**Get Method**

HTTP Request GET /orientation

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /orientation

Request Body:

{“name” : string}

Will return 400 error if orientation does not exist in Database

**Mood**

URIs relative to <http://localhost:8080>

Mood Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /mood

Request Body:

{ “name” : string}

Will Return 400 Error if mood is already in Database

**Get Method**

HTTP Request GET /mood

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /mood

Request Body:

{“name” : string}

Will return 400 error if mood does not exist in Database

**Physical Exam Description (For scalp/brows, head/face, neck, chest, back, abdomen, left/right upper extremity, left/right lower extremity assuming they use the same descriptions)**

URIs relative to <http://localhost:8080>

Mood Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /physical-exam-description

Request Body:

{ “name” : string}

Will Return 400 Error if physical-exam description is already in Database

**Get Method**

HTTP Request GET /physical-exam

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /physical-exam

Request Body:

{“name” : string}

Will return 400 error if physical-exam does not exist in Database

**Edema Description**

URIs relative to <http://localhost:8080>

Edema Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /edema

Request Body:

{ “name” : string}

Will Return 400 Error if edema is already in Database

**Get Method**

HTTP Request GET /edema

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /edema

Request Body:

{“name” : string}

Will return 400 error if edema does not exist in Database

**Other Physical Description**

URIs relative to <http://localhost:8080>

Other Physical Description Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /other-physical-description

Request Body:

{ “name” : string}

Will Return 400 Error if other physical description is already in Database

**Get Method**

HTTP Request GET /other-physical-description

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /other-physical-description

Request Body:

{“name” : string}

Will return 400 error if other physical description does not exist in Database

**Inflammatory**

URIs relative to <http://localhost:8080>

Inflammatory Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /inflammatory

Request Body:

{ “name” : string}

Will Return 400 Error if inflammatory is already in Database

**Get Method**

HTTP Request GET /inflammatory

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /inflammatory

Request Body:

{“name” : string}

Will return 400 error if inflammatory does not exist in Database

**Foot Temperature**

URIs relative to <http://localhost:8080>

Foot Temperature Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /foot-temperature

Request Body:

{ “name” : string}

Will Return 400 Error if foot temperature is already in Database

**Get Method**

HTTP Request GET /foot-temperature

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /foot-temperature

Request Body:

{“name” : string}

Will return 400 error if foot temperature does not exist in Database

**Monofilament**

URIs relative to <http://localhost:8080>

Monofilament Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /monofilament

Request Body:

{ “name” : string}

Will Return 400 Error if monofilament is already in Database

**Get Method**

HTTP Request GET /monofilament

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /monofilament

Request Body:

{“name” : string}

Will return 400 error if monofilament does not exist in Database

**Babinski Reflex**

URIs relative to <http://localhost:8080>

Babinski Reflex Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /babinski-reflex

Request Body:

{ “name” : string}

Will Return 400 Error if Babinski reflex is already in Database

**Get Method**

HTTP Request GET /babinski-reflex

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /babinski-reflex

Request Body:

{“name” : string}

Will return 400 error if babinski reflex does not exist in Database

**Investigation**

URIs relative to <http://localhost:8080>

Investigation Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /investigation

Request Body:

{ “name” : string}

Will Return 400 Error if investigation is already in Database

**Get Method**

HTTP Request GET /investigation

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /investigation

Request Body:

{“name” : string}

Will return 400 error if investigation does not exist in Database

**Etiology**

URIs relative to <http://localhost:8080>

Etiology Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /etiology

Request Body:

{ “name” : string}

Will Return 400 Error if etiology is already in Database

**Get Method**

HTTP Request GET /etiology

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /etiology

Request Body:

{“name” : string}

Will return 400 error if etiology does not exist in Database

**Pressure Stage**

URIs relative to <http://localhost:8080>

Pressure Stage Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /pressure-stage

Request Body:

{ “name” : string}

Will Return 400 Error if pressure stage is already in Database

**Get Method**

HTTP Request GET /pressure-stage

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /pressure-stage

Request Body:

{“name” : string}

Will return 400 error if pressure stage does not exist in Database

**Wound Location**

URIs relative to <http://localhost:8080>

Wound Location Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /wound-location

Request Body:

{ “name” : string}

Will Return 400 Error if wound location is already in Database

**Get Method**

HTTP Request GET /wound-location

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /wound-location

Request Body:

{“name” : string}

Will return 400 error if wound location does not exist in Database

**Wound Description**

URIs relative to <http://localhost:8080>

Wound Description Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /wound-description

Request Body:

{ “name” : string}

Will Return 400 Error if wound description is already in Database

**Get Method**

HTTP Request GET /wound-description

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /wound-description

Request Body:

{“name” : string}

Will return 400 error if wound description does not exist in Database

**Wound Objective**

URIs relative to <http://localhost:8080>

Wound Objective Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /wound-objective

Request Body:

{ “name” : string}

Will Return 400 Error if wound objective is already in Database

**Get Method**

HTTP Request GET /wound- objective

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /wound- objective

Request Body:

{“name” : string}

Will return 400 error if wound objective does not exist in Database

**Exudate**

URIs relative to <http://localhost:8080>

Exudate Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /exudate

Request Body:

{ “name” : string}

Will Return 400 Error if exudate is already in Database

**Get Method**

HTTP Request GET /exudate

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /exudate

Request Body:

{“name” : string}

Will return 400 error if exudate does not exist in Database

**Amount**

URIs relative to <http://localhost:8080>

Amount Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /amount

Request Body:

{ “name” : string}

Will Return 400 Error if amount is already in Database

**Get Method**

HTTP Request GET /amount

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /amount

Request Body:

{“name” : string}

Will return 400 error if amount does not exist in Database

**Other Tissue**

URIs relative to <http://localhost:8080>

Other Tissue Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /other-tissue

Request Body:

{ “name” : string}

Will Return 400 Error if amount is already in Database

**Get Method**

HTTP Request GET /other-tissue

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /other-tissue

Request Body:

{“name” : string}

Will return 400 error if other-tissue does not exist in Database

**Periwound**

URIs relative to <http://localhost:8080>

Periwound Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /periwound

Request Body:

{ “name” : string}

Will Return 400 Error if periwound is already in Database

**Get Method**

HTTP Request GET /periwound

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /periwound

Request Body:

{“name” : string}

Will return 400 error if periwound does not exist in Database

**Blister**

URIs relative to <http://localhost:8080>

Blister Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /blister

Request Body:

{ “name” : string}

Will Return 400 Error if blister is already in Database

**Get Method**

HTTP Request GET /blister

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /blister

Request Body:

{“name” : string}

Will return 400 error if blister does not exist in Database

**No Debridement Reason**

URIs relative to <http://localhost:8080>

No Debridement Reason Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /no-debridement-reason

Request Body:

{ “name” : string}

Will Return 400 Error if no debridement reason is already in Database

**Get Method**

HTTP Request GET /no-debridement-reason

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /no-debridement-reason

Request Body:

{“name” : string}

Will return 400 error if no debridement reason does not exist in Database

**Wound Progress**

URIs relative to <http://localhost:8080>

Wound Progress Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /wound-progress

Request Body:

{ “name” : string}

Will Return 400 Error if wound progress is already in Database

**Get Method**

HTTP Request GET /wound-progress

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /wound-progress

Request Body:

{“name” : string}

Will return 400 error if wound progress does not exist in Database

**Dressings**

URIs relative to <http://localhost:8080>

Dressings Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /dressings

Request Body:

{ “name” : string}

Will Return 400 Error if dressings is already in Database

**Get Method**

HTTP Request GET /dressings

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /dressings

Request Body:

{“name” : string}

Will return 400 error if dressings does not exist in Database

**Frequency**

URIs relative to <http://localhost:8080>

Frequency Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /frequency

Request Body:

{ “name” : string}

Will Return 400 Error if frequency is already in Database

**Get Method**

HTTP Request GET /frequency

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /frequency

Request Body:

{“name” : string}

Will return 400 error if frequency does not exist in Database

**Recommendations**

URIs relative to <http://localhost:8080>

Recommendation Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /recommendation

Request Body:

{ “name” : string}

Will Return 400 Error if recommendation is already in Database

**Get Method**

HTTP Request GET /recommendation

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /recommendation

Request Body:

{“name” : string}

Will return 400 error if recommendation does not exist in Database

**Source Of History**

URIs relative to <http://localhost:8080>

Source Of History Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /source-of-history

Request Body:

{ “name” : string}

Will Return 400 Error if source of history is already in Database

**Get Method**

HTTP Request GET /source-of-history

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /source-of-history

Request Body:

{“name” : string}

Will return 400 error if source of history does not exist in Database

**Risk Benefits**

URIs relative to <http://localhost:8080>

Risk Benefits Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /risk-benefit

Request Body:

{ “name” : string}

Will Return 400 Error if risk benefit is already in Database

**Get Method**

HTTP Request GET /risk-benefit

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /risk-benefit

Request Body:

{“name” : string}

Will return 400 error if risk benefit does not exist in Database

**Debridement Tissue**

URIs relative to <http://localhost:8080>

Debridement Tissue Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /debridement-tissue

Request Body:

{ “name” : string}

Will Return 400 Error if debridement tissue is already in Database

**Get Method**

HTTP Request GET /debridement-tissue

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /debridement-tissue

Request Body:

{“name” : string}

Will return 400 error if debridement tissue does not exist in Database

**Indications**

URIs relative to <http://localhost:8080>

Indication Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /indication

Request Body:

{ “name” : string}

Will Return 400 Error if indication is already in Database

**Get Method**

HTTP Request GET /indication

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /indication

Request Body:

{“name” : string}

Will return 400 error if indication does not exist in Database

**Anesthesia**

URIs relative to <http://localhost:8080>

Anesthesia Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /anesthesia

Request Body:

{ “name” : string}

Will Return 400 Error if anesthesia is already in Database

**Get Method**

HTTP Request GET /anesthesia

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /anesthesia

Request Body:

{“name” : string}

Will return 400 error if anesthesia does not exist in Database

**Instruments**

URIs relative to <http://localhost:8080>

Instrument Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /instrument

Request Body:

{ “name” : string}

Will Return 400 Error if instrument is already in Database

**Get Method**

HTTP Request GET /instrument

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /instrument

Request Body:

{“name” : string}

Will return 400 error if instrument does not exist in Database

**Complications**

URIs relative to <http://localhost:8080>

Complication Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /complication

Request Body:

{ “name” : string}

Will Return 400 Error if complication is already in Database

**Get Method**

HTTP Request GET /complication

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /complication

Request Body:

{“name” : string}

Will return 400 error if complication does not exist in Database

**Dispositions**

URIs relative to <http://localhost:8080>

Disposition Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /disposition

Request Body:

{ “name” : string}

Will Return 400 Error if disposition is already in Database

**Get Method**

HTTP Request GET /disposition

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /disposition

Request Body:

{“name” : string}

Will return 400 error if disposition does not exist in Database

**Cauterization Options**

URIs relative to <http://localhost:8080>

Cauterization Option Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /cauterization-option

Request Body:

{ “name” : string}

Will Return 400 Error if cauterization option is already in Database

**Get Method**

HTTP Request GET /cauterization-option

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /cauterization-option

Request Body:

{“name” : string}

Will return 400 error if cauterization option does not exist in Database

**Supplementation Ordered**

URIs relative to <http://localhost:8080>

Supplementation Ordered Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /supplementation-ordered

Request Body:

{ “name” : string}

Will Return 400 Error if supplementation ordered is already in Database

**Get Method**

HTTP Request GET /supplementation-ordered

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /supplementation-ordered

Request Body:

{“name” : string}

Will return 400 error if supplementation ordered does not exist in Database

**Discussed With**

URIs relative to <http://localhost:8080>

Discussed With Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /discussed-with

Request Body:

{ “name” : string}

Will Return 400 Error if discussed with is already in Database

**Get Method**

HTTP Request GET /discussed-with

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /discussed-with

Request Body:

{“name” : string}

Will return 400 error if discussed with does not exist in Database

**Information**

URIs relative to <http://localhost:8080>

Information Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /information

Request Body:

{ “name” : string}

Will Return 400 Error if information is already in Database

**Get Method**

HTTP Request GET /information

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /information

Request Body:

{“name” : string}

Will return 400 error if information does not exist in Database

**Test Types**

URIs relative to <http://localhost:8080>

Test Types Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /test-type

Request Body:

{ “name” : string}

Will Return 400 Error if test type is already in Database

**Get Method**

HTTP Request GET /test-type

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /test-type

Request Body:

{“name” : string}

Will return 400 error if test type does not exist in Database

**Follow Up Range**

URIs relative to <http://localhost:8080>

Follow Up Range Representation

{

“name”: *string*

}

**Create Method**

HTTP Request: POST /follow-up-range

Request Body:

{ “name” : string}

Will Return 400 Error if source of history is already in Database

**Get Method**

HTTP Request GET /follow-up-range

Response:

[

{

“name”:string

},

{

“name”:string

},

…

{

“name”:string

}

]

**Delete Method**

HTTP Request: DELETE /follow-up-range

Request Body:

{“name” : string}

Will return 400 error if follow up range does not exist in Database

**Patients**

URIs relative to <http://localhost:8080>

Quick Patient Representation

{

“id” : *int,*

“firstName”: *string,*

“lastName” : *string,*

“lastVisitDate” : *date*,

“followUpVisitDate” : *date*,

“isActive” : *Boolean*,

“location”: *string*

}

**Get Method**

**Get Patients By Location**

HTTP Request GET /patient?{location}

Required Query Parameter:

Location

Response:

Returns all Patient Resources For The Given Location

Error if:

-location does not exist in application

**Create Method**

HTTP Request: POST /patient

Request Body:

Valid Patient Resource

Response:

If successful, this method returns a Patient Resource

Error If:

Will Return 400 Error if required data in Patient Resource is missing

**Update Method**

HTTP Request: PUT /patient/{id}

Request Body:

Valid Patient Resource

Response:

If successful, this method returns a Patient Resource

Error If:

Will Return 4xx Error if required data in Patient Resource is missing

Will Return 4xx Error if Patient Resource does not exist

**Delete Method**

HTTP Request: DELETE /patient/{id}

Request Body:

Do not send a request body with this method

Response Body:

Empty response body if successful

Error If:

Will return 400 error if id does not exist

**Wounds**

URIs relative to <http://localhost:8080>

Quick Wound Representation

{

“id” : *int,*

“firstName”: *string,*

“lastName” : *string,*

“lastVisitDate” : *date*,

“followUpVisitDate” : *date*,

“isActive” : *Boolean*,

“location”: *string*

}

**Get Method**

**Get Patient Wounds**

HTTP Request GET /patient/{id}/wound

Request Body:

No request body needed

Response:

Returns all Wound Resources For The Given Patient Resource

Error if:

-Patient id does not exist

**Create Method**

HTTP Request: POST /patient/{id}/wound

Request Body:

Valid Wound Resource

Response:

If successful, this method returns a Wound Resource

Error If:

Will Return 400 Error if required data in Wound Resource is missing or incorrect

**Update Method**

HTTP Request: PUT /patient/{id}/wound/{id}

Request Body:

Valid Wound Resource

Response:

If successful, this method returns a Wound Resource

Error If:

Will Return 4xx Error if required data in Wound Resource is missing or incorrect

Will Return 4xx Error if Wound does not exist

**Delete Method**

HTTP Request: DELETE /patient/{id}/wound

Request Body:

Do not send a request body with this method

Response Body:

Empty response body if successful

Error If:

Will return 400 error if id does not exist

**Wound Debridements**

URIs relative to <http://localhost:8080>

Quick Debridement Representation

{

“id” : *int,*

“firstName”: *string,*

“lastName” : *string,*

“lastVisitDate” : *date*,

“followUpVisitDate” : *date*,

“isActive” : *Boolean*,

“location”: *string*

}

**Get Method**

**Get Patient Wounds**

HTTP Request GET /patient/{id}/wound/{id}/debridement

Request Body:

No request body needed

Response:

Returns all Debridement Resources For The Given Wound Resource

Error if:

-Patient id does not exist

-Wound id does not exist

**Create Method**

HTTP Request: POST /patient/{id}/wound/{id}/debridement

Request Body:

Valid Debridement Resource

Response:

If successful, this method returns a Debridement Resource

Error If:

Will Return 400 Error if required data in Debridement Resource is missing or incorrect

Will return 4xx Error if Patient Id does not exist

Will return 4xx Error if Wound Id does not exist

**Update Method**

HTTP Request: PUT /patient/{id}/wound/{id}/debridement/{id}

Request Body:

Valid Debridement Resource

Response:

If successful, this method returns a Debridement Resource

Error If:

Will Return 4xx Error if required data in Debridement Resource is missing or incorrect

Will Return 4xx Error if Patient does not exist

Will Return 4xx Error if Wound does not exist

Will Return 4xx Error if Debridement does not exist

**Delete Method**

HTTP Request: DELETE /patient/{id}/wound/{id}/debridement/{id}

Request Body:

Do not send a request body with this method

Response Body:

Empty response body if successful

Error If:

Will return 400 error if Patient does not exist

Will return 4xx error if Wound does not exist

Will return 4xx error if Debridement does not exist

**Physical Exams**

URIs relative to <http://localhost:8080>

Quick Physical Exam Representation

{

“id” : *int,*

“firstName”: *string,*

“lastName” : *string,*

“lastVisitDate” : *date*,

“followUpVisitDate” : *date*,

“isActive” : *Boolean*,

“location”: *string*

}

**Get Method**

**Get Physical Exams**

HTTP Request GET /patient/{id}/physical-exam

Request Body:

No request body needed

Response:

Returns all Physical Exam Resources For The Given Patient Resource

Error if:

-Patient id does not exist

**Create Method**

HTTP Request: POST /patient/{id}/physical-exam

Request Body:

Valid Physical Exam Resource

Response:

If successful, this method returns a Physical Exam Resource

Error If:

Will Return 400 Error if required data in Physical Exam Resource is missing or incorrect

**Update Method**

HTTP Request: PUT /patient/{id}/physical-exam/{id}

Request Body:

Valid Physical Exam Resource

Response:

If successful, this method returns a Physical Exam Resource

Error If:

Will Return 4xx Error if required data in Physical Exam Resource is missing or incorrect

Will Return 4xx Error if Patient does not exist

Will Return 4xx Error if Physical Exam does not exist

**Delete Method**

HTTP Request: DELETE /patient/{id}/physical-exam/{id}

Request Body:

Do not send a request body with this method

Response Body:

Empty response body if successful

Error If:

Will Return 4xx Error if Patient does not exist

Will Return 4xx Error if Physical Exam does not exist

**Tests**

URIs relative to <http://localhost:8080>

Quick Test Representation

{

“id” : *int,*

“firstName”: *string,*

“lastName” : *string,*

“lastVisitDate” : *date*,

“followUpVisitDate” : *date*,

“isActive” : *Boolean*,

“location”: *string*

}

**Get Method**

**Get Tests**

HTTP Request GET /patient/{id}/test

Request Body:

No request body needed

Response:

Returns all Test Resources For The Given Patient Resource

Error if:

-Patient id does not exist

**Create Method**

HTTP Request: POST /patient/{id}/test

Request Body:

Valid Test Resource

Response:

If successful, this method returns a Test Resource

Error If:

Will Return 400 Error if required data in Test Resource is missing or incorrect

**Update Method**

HTTP Request: PUT /patient/{id}/test/{id}

Request Body:

Valid Test Resource

Response:

If successful, this method returns a Test Resource

Error If:

Will Return 4xx Error if required data in Test Resource is missing or incorrect

Will Return 4xx Error if Patient does not exist

Will Return 4xx Error if Test does not exist

**Delete Method**

HTTP Request: DELETE /patient/{id}/test/{id}

Request Body:

Do not send a request body with this method

Response Body:

Empty response body if successful

Error If:

Will Return 4xx Error if Patient does not exist

Will Return 4xx Error if Test does not exist

**Face Sheets**

URIs relative to <http://localhost:8080>

Quick Face Sheet Representation

{

“id” : *int,*

“fileLocation”: *string,*

}

**Get Method**

**Get Tests**

HTTP Request GET /patient/{id}/face-sheet

Request Body:

No request body needed

Response:

Returns all Face Sheet Resources For The Given Patient Resource

Error if:

-Patient id does not exist

**Create Method**

HTTP Request: POST /patient/{id}/face-sheet

Request Body:

Valid Test Resource

Response:

If successful, this method returns a Test Resource

Error If:

Will Return 400 Error if required data in Test Resource is missing or incorrect

**Delete Method**

HTTP Request: DELETE /patient/{id}/face-sheet/{id}

Request Body:

Do not send a request body with this method

Response Body:

Empty response body if successful

Error If:

Will Return 4xx Error if Patient does not exist

Will Return 4xx Error if Test does not exist

**Attached Document**

URIs relative to <http://localhost:8080>

Quick Attached Document Representation

{

“id” : *int,*

“fileLocation”: *string,*

}

**Get Method**

**Get Tests**

HTTP Request GET /patient/{id}/attached-document

Request Body:

No request body needed

Response:

Returns all Attached Document Resources For The Given Patient Resource

Error if:

-Patient id does not exist

**Create Method**

HTTP Request: POST /patient/{id}/attached-document

Request Body:

Valid Test Resource

Response:

If successful, this method returns a Test Resource

Error If:

Will Return 400 Error if required data in Test Resource is missing or incorrect

**Delete Method**

HTTP Request: DELETE /patient/{id}/attached-document/{id}

Request Body:

Do not send a request body with this method

Response Body:

Empty response body if successful

Error If:

Will Return 4xx Error if Patient does not exist

Will Return 4xx Error if Attached Document does not exist