API Documentation for Bayer Patient Finder Application

This document contains the backend for the Patient Finder Application. The auth-token is a unique token a user may receive after they get logged in, and it expires after 1 day duration from the time when the user logs in.

User-Specific API

Method	Route URI	Request Body/ Query (JSON)	Response Body	Details

PUT	/user
	
	/user/login
		{
		userid,
		password
		}
			{
			success: isLogged,
			userData: {
			userid,
			fullName,
			email,
			authToken
			},
			message
			}
				Provides a way for a user to gain access to the patient finder database. Start logging user's accesses for the database by the using of an unique authToken. (Security feature)
				
				authToken expires after 1 day.

POST	/user
	
	/user/register
		{
		userid,
		password,
		fullname,
		email
		}
			{
			success: isLogged,
			userData: {
			userid,
			fullName,
			email,
			authToken
			},
			message
			}
				Create a new user to the patient finder application and provides a way for a user to gain access to the patient finder database. Start logging user's accesses for the database by the using of an unique authToken. (Security feature)
				
				authToken expires after 1 day.

PUT	/user/logout	{
		userid,
		authToken
		}
			{
			success: !isLogged,
			message
			}
				Users will be logged out and the session will be ended. The user, upon logout, will no longer have their access until a new accessToken is generated by logging in.

GET	/users/preferences	{
		userid,
		authToken
		}
			{
			success,
			defaultPreferenceId,
			preferenceData:[
			userPref1:{
			userid,
			id, // preferenceId
			savedName,
			jsonData,
			createdAt
			},
			userPref2
			userPref3
				Get a list of preferences, specific to a user with userid.
				
				jsonData contains the filter values stored at mySQL end.
				More details on jsonData format is provided in the jsonData Format section
				
				Note: when a user is new to the PF system their

			… …] }	*default Preference is none (NULL), until they create their first preference.*

POST	/users/preferences	{
		userid,
		authToken,
		saveName,
		jsonData,
		makeDefault:bool
		}
			{
			success,
			message,
			preferenceId
			}
				Post/ Create a new preference with name, saveName, for user with userid.
				
				jsonData contains the filter values.
				More details on jsonData format is provided in the jsonData Format section

PUT	/users/preferences	{
		userid,
		auth-token,
		preferenceId,
		jsonData
		}
			{
			success,
			message,
			preferenceId,
			createdAt
			}
				Put/ Update/ Edit an existing preference identified by preferenceId belonging to a user identified by userid.
				
				jsonData contains the filter values.
				More details on jsonData format is provided in the jsonData Format section

DELETE	/users/preferences	{
		userid,
		auth-token,
		preferenceId
		}
			{
			success,
			message,
			preferenceId
			}
				Delete the preference record identified by preferenceId belonging to a user identified by userid.

GET	/users/history	{
		userid,
		auth-token,
		rangeSeq:{
		start:1, #default
		end:50
		}
		}
			{
			userid,
			sequence:{
			start, end
			}
			historyData:[
			userHistory1:{
			historyId,
			createdOn.
			jsonData
			},
			userHistory2
			… ...
			]
			}
				Get the history of patient finder data access by a user identified with userid.

POST	/users/history	{
		userid,
		auth-token,
		jsonData
		}
			{
			success,
			message
			}
				Create a record for user Patient finder data access for generating graphs using filter values present inside jsonData.

Patient Finder Database API

Method	Route URI	Request Body/ Query (JSON)	Response Body	Details

GET	/patientfinder/labels	{
		userid,
		authToken
		}
			{
			success,
			labelData:[
			<labelrecord1>,
			< labelrecord2>,
			… …
			… ...
			]
			}
				Get all labels, values and their names from label_info that are either of type medical condition or treatments
				
				labelrecordi is a dictionary.
				{
				name, label, label_type, label_val
				}

GET	/patientfinder/values/states	{
		userid,
		authToken
		}
			{
			success,
			stateData:[
			<state1>,
			<state2>,
			…. ….
			…. ….
			]
			}
				Get all possible states from patient_info.
				
				state[i] is a dictionary.
				{
				state: value
				}

GET	/patientfinder/values/paytyp	{
		userid,
		authToken
		}
			{
			success,
			paytypData:[
			<paytyp1>,
			<paytyp2>,
			…. ….
			…. ….
			]
			}
				Get all possible payer types from patient_info.
				
				paytyp[i] is a dictionary.
				{
				paytyp: value
				}

GET	/patientfinder/values/cohort	{
		userid,
		authToken
		}
			{
			success,
			popData:[
			<pop1>,
			<pop2>,
			…. ….
			…. ….
			]
			}
				Get all possible payer types from patient_info.
				
				pop[i] is a dictionary.
				{
				pop: value
				}

POST	/patientfinder/treatments	{
		userid,
		authToken,
		jsonData
		}
			{
			success,
			group_condition,
			treatments: {
			labels,
			data
			}
			}
				Generate the PatientFinder data required for data visualization (graph) purpose. For treatments only.
				
				jsonData contains filter values
				More details on jsonData format is provided in the jsonData Format section

				(POST method used with safe or no database edit functionality)

POST	/patientfinder/medicals	{
		userid,
		auth-token,
		jsonData
		}
			{
			success,
			group_condition,
			medical_conditions:{
			labels,
			data
			}
			}
				Generate the PatientFinder data required for data visualization (graph) purpose. For medicals only.
				
				jsonData contains filter values
				More details on jsonData format is provided in the jsonData Format section
				
				(POST method used with safe or no database edit functionality)

jsonData Format:

{
group_condition: {
group_by: “cohort” | “paytype”,
selection: [(cohort | paytype) group_by values of Array form]
},
states: [list of all states where patients data is queried from],
treatments: {
labels : [list of treatment labels which are specifically focused],
OR: [
list of label values, for which either one of the labels need to exist in the selected patient record
],
AND: [list of label values, all of the labels must exist in the selected patient records]
}
medical_conditions: {
labels : [list of medical_conditions labels which are specifically focused],
OR: [
list of label values, for which either one of the labels need to exist in the selected patient record
],
AND: [list of label values, all of the labels must exist in the selected patient records]
}
}