MenuWorld API

Method	Endpoint	Description	Input JSON	Output JSON
POST*	/users	Create user	email, userName, password, owner (boolean)	userID, token
GET*	/users/:uid	Get user's name, status, and likes	Nothing	userID, userName, owner (boolean), foods (array of FOODs)
GET	/users/:uid/private	Get user's email, recently-viewed meals, and menus (owners only)	{ } (empty JSON)	email, foods (array of foodID), menus (array of {menuID, restaurantName, menuName, address})
DELETE	/users/:uid	Suspend user (can be called by the user himself/ herself or admin)	{ } (empty JSON)	Just an HTTP OK
PUT		Change password and/or email	oldPassword, [password], [email] (square brackets mean "optional")	Just an HTTP OK
POST	/login	Log in	email, password	userID, token
POST	/menus	Create menu	restaurantName, menuName, address	menuID
GET*	/menus/:mid	Get menu	Nothing	menuID, restaurantName, menuName, address, foods (array of FOODs)
DELETE		Delete menu	{ } (empty JSON)	Just an HTTP OK
PUT		Modify menu	[restaurantName], [menuName], [address]	Just an HTTP OK

FOOD: foodID, foodName, menuID, photo (in base 64 encoding), likes (array of userID strings), mealType (string), cuisine (string), allergens (array of ARs)

To determine if the current user likes the food, you can store the userID in client local storage and match the userID. Use GET users/:uid to find out who the users are.

Method	Endpoint	Description	Input JSON	Output JSON
POST	/foods	Add food to menu	menuID, foodName, [photo (in base 64 encoding)], mealType (string), cuisine (string), allergens (array of strings)	foodID
GET*	/foods/:fid[/view] (if "/view", the food will be registered in the user's view history)	Get food details	Nothing	FOOD
PUT	/foods/:fid	Modify food	[foodName], [photo (in base 64 encoding)], [mealType (string)], [cuisine (string)], [allergens (array of AR)]	Just an HTTP OK
DELETE		Delete food	{ } (empty JSON)	
POST	/foods/:fid/allergen	Report (confirm/deny) allergen	allergen (string), confirm (boolean)	
DELETE		Un-report allergen	allergen (string)	
POST	/foods/:fid/like	Like food	{ } (empty JSON)	
DELETE		Un-like food	{ } (empty JSON)	
GET*	/top	Top searches	Nothing	searches (array of strings)
	/search/:search+term/:allergy1+allergy2+allergy3+ …	Search for foods (search terms are fuzzy-matched; allergies, which can only be one word each, are exactly matched)	Nothing	foods (array of FOODs)
	/popular	Popular (i.e. most liked) foods	Nothing	foods (array of FOODs)
	/latest	Most recently-posted foods	Nothing	

* Token not required. Other requests: pass token via the **bearer authorization** HTTP header.

AR: allergen (string), confirms (array of userIDs), denys (array of userIDs)

"Allergen" means that the food does **not** contain the ingredient listed.