MenuWorld API

Endpoint	Method	Description	Input JSON	Output JSON	Possible errors	Scope
/users	POST*	Create user	email, userName, password, owner (boolean)	userID, token	400: missing required input 403: duplicate email or username	All
/users/:uid	GET*	Get user’s name, status, and likes	Nothing	userID, userName, owner (boolean), foods (array of FOODs)	None	
/users/:uid/private	GET	Get user’s email, recently-viewed meals, and menus (owners only)	Nothing	email, foods (array of foodID), menus (array of {menuID, restaurantName, menuName, address})	401: unauthorized (uid doesn’t match that given by token)	Any logged-in user
/users/:uid	DELETE	Suspend user (can be called by the user himself/ herself or admin)	{ } (empty JSON)	Just an HTTP OK		Any logged-in user and admin
	PUT	Change password and/or email	oldPassword, [password], [email] (square brackets mean “optional”)	Just an HTTP OK	400: missing required input 401: unauthorized 403: try another password or email	Any logged-in user or admin
/login	POST*	Log in	email, password	userID, token	400: missing required input 403: login failed	All
/menus	POST	Create menu	restaurantName, menuName, address	menuID	400: missing required input 401: not an owner	Logged-in user (owners only)

Endpoint	Method	Description	Input JSON	Output JSON	Possible errors	Scope
/menus/:mid	GET*	Get menu	Nothing	menuID, restaurantName, menuName, address, foods (array of FOODs)	403: invalid menuID	All
	DELETE	Delete menu	{ } (empty JSON)	Just an HTTP OK		Logged-in user (owners only)
	PUT	Modify menu	[restaurantName], [menuName], [address]	Just an HTTP OK	400: missing required input 403: invalid menuID	
/foods	POST	Add food to menu	menuID, foodName, [photo (in base 64 encoding)], mealType (string), cuisine (string), allergens (array of strings)	foodID	400: missing required input 403: invalid menuID	
/foods/:fid[/view] (if "/view", the food will be registered in the user's view history)	GET*	Get food details	Nothing	FOOD	403: invalid foodID	All
/foods/:fid	PUT	Modify food	[foodName], [photo (in base 64 encoding)], [mealType (string)], [cuisine (string)], [allergens (array of AR)]	Just an HTTP OK	400: missing required input 403: invalid foodID	Logged-in user (owners only)
	DELETE	Delete food	{ } (empty JSON)		403: invalid foodID	

Endpoint	Method	Description	Input JSON	Output JSON	Possible errors	Scope
/foods/:fid/allergen	POST	Report (confirm/ deny) allergen	allergen (string), confirm (boolean)	Just an HTTP OK	400: missing required input 403: invalid allergen, or it's been already been (un-)reported	Any logged-in user
	DELETE	Un-report allergen	allergen (string)			
/foods/:fid/like	POST	Like food	{ } (empty JSON)		403: food already (un-)liked	
	DELETE	Un-like food	{ } (empty JSON)			
/top	GET*	Top searches	Nothing	searches (array of strings)	None	All
/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			

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

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

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.

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