BE services	Request	Response
Authentication	/user: post: tags: - user summary: Creates a new user. description: Creates a new user account. operationId: createUser requestBody: description: The request body for creating a new user. It should contain all necessary user details like username, email, and password. The username and email must be unique. content: application/json: schema: $ref: '#/components/schemas/User'	responses: '201': description: User successfully created content: application/json: schema: $ref: '#/components/schemas/User' '400': description: Invalid input '401': description: Unauthorized. Authentication is required. '409': description: Conflict. User already exists.
	/user/{userId}: put: tags: - user summary: Updates a specific user's details. description: This can only be done by the logged in user. operationId: updateUser parameters: - name: userId in: path description: name that need to be deleted required: true schema: type: integer format: int64 requestBody: description: The request body for updating an existing user. It should contain all the fields that need to be updated for the specified user. Fields that are not included in the request will remain unchanged. content: application/json: schema: $ref: '#/components/schemas/User'	responses: '200': description: User successfully updated content: application/json: schema: $ref: '#/components/schemas/User' '400': description: Invalid input '401': description: Unauthorized. Authentication is required. '404': description: User not found
	/user/login: get: tags: - user summary: Logs user into the system. description: Logs a specific user into the system	responses: '200': description: successful operation headers: X-Rate-Limit: description: calls per hour allowed by the user

	operationId: loginUser parameters: - name: username in: query description: The username for login required: false schema: type: string - name: password in: query description: The password for login in clear text required: false schema: type: string	schema: type: integer format: int32 X-Expires-After: description: date in UTC when token expires schema: type: string format: date-time content: application/xml: schema: type: string application/json: schema: type: string '400': description: Invalid username/password supplied
	/user/logout: get: tags: - user summary: Logs out current logged in user session. description: Logs a specific user out of the system operationId: logoutUser parameters: []	responses: default: description: successful operation
	/user/{userId}: get: tags: - user summary: Gets user details by userId. description: '' operationId: getUserByName parameters: - name: userId in: path description: 'The name that needs to be fetched. Use user1 for testing. ' required: true schema: type: integer format: int64	responses: '200': description: successful operation content: application/json: schema: $ref: '#/components/schemas/User' application/xml: schema: $ref: '#/components/schemas/User' '400': description: Invalid username supplied '404': description: User not found
	/user/{userId}: delete: tags: - user summary: Deletes a specific user's account. description: Deletes a specific user account.	responses: '400': description: Invalid username supplied '404': description: User not found

	operationId: deleteUser parameters: - name: userId in: path description: The name that needs to be deleted required: true schema: type: integer format: int64	
Product	/items: post: tags: - product summary: Add a new product. description: Add a new product to the app. operationId: addItem requestBody: description: The request body for creating a new map. It should contain all necessary user details like name, image, price, details. content: application/json: schema: $ref: '#/components/schemas/Product' required: true	responses: '201': description: Product successfully added content: application/json: schema: type: array items: $ref: '#/components/schemas/Product' '400': description: Invalid input '422': description: Validation exception
	/items/{itemId}: put: tags: - product summary: Updates a specific product's information. description: '' operationId: updateItemWithForm parameters: - name: itemId in: path description: ID of product that needs to be updated required: true schema: type: integer format: int64 requestBody: description: The request body for updating an existing product. It should contain all the fields that need to be updated for the specified product (name, image, price, details). Fields that are not included in the request will remain unchanged. content: application/json: schema:	responses: '200': description: The product was successfully updated. content: application/json: schema: $ref: '#/components/schemas/Product' '400': description: Invalid ID supplied '404': description: Product not found

	$ref: '#/components/schemas/Product' required: true	
	/items/{itemId}/ratings: post: tags: - product summary: Adds a rating to a product. description: '' operationId: putsRatings parameters: - name: itemId in: path description: ID of product to rate required: true schema: type: integer format: int64 requestBody: description: The request body for creating a new rating. It should contain all necessary rating details like user, rating (1 to 5). content: application/json: schema: $ref: '#/components/schemas/Ratings' required: true	responses: '201': description: Rating successfully added content: application/json: schema: $ref: '#/components/schemas/Product' '400': description: Invalid input '404': description: Product not found
	/items/{itemId}/ratings/{ratingId}: put: tags: - product summary: Updates a product's specific rating. description: '' operationId: updateRatingWithForm parameters: - name: itemId in: path description: ID of product which rating needs to be updated required: true schema: type: integer format: int64 - name: ratingId in: path description: Id of the specific Rating to update required: true schema: type: integer format: int64 requestBody: description: The request body for updating an existing rating. It should contain all the fields that	responses: '200': description: The rating was successfully updated. content: application/json: schema: $ref: '#/components/schemas/Product' '400': description: Invalid input '404': description: Product not found

	need to be updated for the specified rating (user,rating). Fields that are not included in the request will remain unchanged. content: application/json: schema: $ref: '#/components/schemas/Ratings' required: true	
	/items/{itemId}: get: tags: - product summary: Find a specific product. description: Returns a single product by ID operationId: getItemById parameters: - name: itemId in: path description: ID of product to return required: true schema: type: integer format: int64	responses: '200': description: successful operation content: application/json: schema: $ref: '#/components/schemas/Product' '400': description: Invalid ID supplied '404': description: Product not found
	/items/{itemId}/ratings: get: tags: - product summary: Gets all ratings from a specific product. description: '' operationId: getsRatings parameters: - name: itemId in: path description: ID of the product required: true schema: type: integer format: int64	responses: '200': description: successful operation content: application/json: schema: $ref: '#/components/schemas/Product' '400': description: Invalid input '404': description: Product not found
	/items/{itemId}: delete: tags: - product summary: Deletes a specific product. description: Deletes a product by ID. operationId: deleteItem parameters: - name: itemId in: path description: Product id to delete	responses: '200': description: Product successfully deleted '400': description: Invalid product value '404': description: Product not found

	required: true schema: type: integer format: int64	
	/items/{itemId}/ratings: delete: tags: - product summary: Deletes all ratings from a specific product. description: Delete every rating that one specific product has. operationId: deleteRatings parameters: - name: itemId in: path description: ID of product which ratings needs to be deleted required: true schema: type: integer format: int64	responses: '200': description: All ratings from that product were successfully deleted content: application/json: schema: $ref: '#/components/schemas/Product' '400': description: Invalid input '404': description: Product not found
	/items/{itemId}/ratings/{ratingId}: delete: tags: - product summary: Deletes a specific product's rating. description: Delete one specific product's rating. operationId: deleteRating parameters: - name: itemId in: path description: ID of product which rating needs to be deleted required: true schema: type: integer format: int64 - name: ratingId in: path description: Id of the specific rating to delete required: true schema: type: integer format: int64	responses: '200': description: Product's rating deleted '400': description: Invalid product or rating value '404': description: Product or Rating IDs not found
Favorites	/favorites/{userId}: get: tags:	responses: '200': description: Successful operation

	- favorites summary: Get all products that a user stored in their favorites. description: Get all user's existing favorites by Id operationId: getFavorites parameters: - name: userId in: path description: ID of the user required: true schema: type: integer format: int64	content: application/json: schema: $ref: '#/components/schemas/Favorites' '400': description: Invalid ID supplied '404': description: Favorites not found '422': description: Validation exception
	/favorites/{userId}: post: tags: - favorites summary: Adds a product to the user's favorites. description: Adds a new product to the user's favorites operationId: addFavorites parameters: - name: userId in: path description: User's id required: true schema: type: integer format: int64 requestBody: description: The request body for adding an existing product to favorites. It should contain the ID of the existing product. content: application/json: schema: $ref: '#/components/schemas/Product' required: true	responses: '201': description: Item successfully added to favorites content: application/json: schema: $ref: '#/components/schemas/Favorites' '400': description: Invalid input '404': description: Product not found '422': description: Validation exception
	/favorites/{userId}/{itemId}: delete: tags: - favorites summary: Deletes a specific product from favorites. description: Deletes a product from favorites by id operationId: deleteFavorite parameters: - name: userId	responses: '200': description: Product Deleted from favorites '400': description: Invalid product value '404': description: Product not found

	in: path description: User id to delete favorites' item required: true schema: type: integer format: int64 - name: itemId in: path description: ID of product that needs to be deleted from user's favorites required: true schema: type: integer format: int64	
Supermarket	/supermarkets: post: tags: - supermarket summary: Add a new supermarket to the app. description: Add a new supermarket to the app operationId: addSupermarket requestBody: description: The request body for creating a new supermarket. It should contain all necessary supermarket details like name, categories, and location. content: application/json: schema: $ref: '#/components/schemas/Supermarket' required: true description: Validation exception	responses: '201': description: Supermarket successfully added content: application/json: schema: $ref: '#/components/schemas/Supermarket' '400': description: Invalid input '422': description: Validation exception
	/supermarkets/{supermarketId}: put: tags: - supermarket summary: Updates a specific supermarket information. description: '' operationId: updateSupermarketWithForm parameters: - name: supermarketId in: path description: ID of supermarket that needs to be updated required: true schema: type: integer format: int64 requestBody: description: The request body for updating an	responses: '200': description: Supermarket successfully updated content: application/json: schema: $ref: '#/components/schemas/Supermarket' '400': description: Invalid ID supplied '404': description: Supermarket not found

	existing supermarket. It should contain all the fields that need to be updated for the specified supermarket (name, categories, or location). Fields that are not included in the request will remain unchanged. content: application/json: schema: $ref: '#/components/schemas/Supermarket' required: true	
	/supermarkets/{supermarketId}/categories: post: tags: - supermarket summary: Add a new categorie to a specific supermarket. description: " operationId: uploadSupermarketCategories parameters: - name: supermarketId in: path description: ID of supermarket required: true schema: type: integer format: int64 requestBody: description: The request body for creating a new categorie. It should contain all necessary categorie details like name and products. content: application/json: schema: $ref: '#/components/schemas/Category' required: true	responses: '201': description: Supermarket category successfully added content: application/json: schema: $ref: '#/components/schemas/Supermarket' '400': description: Invalid ID supplied '404': description: Supermarket not found
	/supermarkets/{supermarketId}/categories/{categoryId}: put: tags: - supermarket summary: Updates a category within a specific supermarket description: Updates an existing category details in the specified supermarket operationId: updateSupermarketCategory parameters: - name: supermarketId in: path description: ID of the supermarket containing the category to update	responses: '200': description: successful operation content: application/json: schema: $ref: '#/components/schemas/Supermarket' '400': description: Invalid ID supplied '404': description: Supermarket not found

	required: true schema: type: integer format: int64 - name: categoryId in: path description: ID of the category to update required: true schema: type: integer format: int64 requestBody: description: The request body for updating an existing category. It should contain all the fields that need to be updated for the specified category (name, products). Fields that are not included in the request will remain unchanged. content: application/json: schema: $ref: '#/components/schemas/Category' required: true	
	/supermarkets/{supermarketId}/categories/{categoryId}/products: put: tags: - supermarket summary: Updates products within a specific category in a specific supermarket. description: Updates existing products in the specified category within a specific supermarket operationId: updateSupermarketCategoryProducts parameters: - name: supermarketId in: path description: ID of the supermarket containing the category of the product to update required: true schema: type: integer format: int64 - name: categoryId in: path description: ID of the category to update required: true schema: type: integer format: int64 requestBody: description: The request body for updating an existing product. It should contain all the fields that need to be updated for the specified product (name,	responses: '200': description: successful operation content: application/json: schema: $ref: '#/components/schemas/Supermarket' '400': description: Invalid ID supplied '404': description: Supermarket not found

	image, price or details). Fields that are not included in the request will remain unchanged. content: application/json: schema: $ref: '#/components/schemas/Product' required: true	
	/supermarkets: get: tags: - supermarket summary: Finds all supermarkets. description: Returns all supermarkets in the app. operationId: findAllSupermarkets	responses: '200': description: successful operation content: application/json: schema: type: array items: $ref: '#/components/schemas/Supermarket' '400': description: Invalid status value
	/supermarkets/{supermarketId}: get: tags: - supermarket summary: Find a specific supermarket. description: Returns a single supermarket by id. operationId: getSupermarketById parameters: - name: supermarketId in: path description: ID of supermarket to return required: true schema: type: integer format: int64	responses: '200': description: successful operation content: application/json: schema: $ref: '#/components/schemas/Supermarket' '400': description: Invalid ID supplied '404': description: Supermarket not found
	/supermarkets/{supermarketId}/categories: get: tags: - supermarket summary: Retrieve all categories from a specific supermarket. description: Returns all categories available at a specific supermarket based on the supermarket ID. operationId: getSupermarketCategories parameters: - name: supermarketId in: path description: ID of the supermarket to retrieve categories from required: true	responses: '200': description: successful operation content: application/json: schema: $ref: '#/components/schemas/Category' '400': description: Invalid ID supplied '404': description: Categories not found

	schema: type: integer format: int64	
	supermarkets/{supermaketId}/categories/{categoryId}/products: get: tags: - supermarket summary: Retrieve a list of products within a specific category in a specific supermarket. description: Returns a list of products in the specified category within a specified supermarket operationId: getProductsInCategory parameters: - name: supermarketId in: path description: ID of the supermarket containing the category of products to retrieve required: true schema: type: integer format: int64 - name: categoryId in: path description: ID of the category to retrieve products from required: true schema: type: integer format: int64	responses: '200': description: Successful operation; list of products in the category content: application/json: schema: type: array items: $ref: '#/components/schemas/Product' '400': description: Invalid status value '404': description: Supermarket not found
	/supermarkets/{supermarketId}: delete: tags: - supermarket summary: Deletes a specific supermarket. description: Delete a specific supermarket by Id. operationId: deleteSupermarket parameters: - name: supermarketId in: path description: Id from the supermarket to delete required: true schema: type: integer format: int64	responses: '200': description: Supermarket was sucessfully deleted '400': description: Invalid supermarket value '404': description: Supermarket not found
	/supermarkets/{supermarketId}/categories: delete: tags:	responses: '200': description: Supermarket's categories

	- supermarket summary: Deletes all categories from a specific supermarket. description: Deletes all categories from the supermarket with the specified ID operationId: deleteSupermarketCategories parameters: - name: supermarketId in: path description: ID of the supermarket whose categories should be deleted required: true schema: type: integer format: int64	sucessfully deleted '400': description: Invalid supermarket's category value '404': description: Supermarket not found
	/supermarkets/{supermarketId}/categories/{categoryId}: delete: tags: - supermarket summary: Deletes a specific category within a specific supermarket. description: Deletes a specific category in the specified supermarket based on the provided supermarket ID and category ID operationId: deleteSupermarketCategory parameters: - name: supermarketId in: path description: ID of the supermarket whose categories should be deleted required: true schema: type: integer format: int64 - name: categoryId in: path description: ID of the category to delete required: true schema: type: integer format: int64 description: Supermarket not found	responses: '200': description: Supermarket's category was Deleted '400': description: Invalid supermarket's category value '404':
	/supermarkets/{supermaketId}/categories/{categoryId}/products: delete: tags: - supermarket summary: Deletes products within a specific category in a specific supermarket. description: Deletes products within the	responses: '200': description: Supermarket's product was Deleted '400': description: Invalid supermarket's category value '404':

	specified category in a specified supermarket operationId: deleteSupermarketCategoryProducts parameters: - name: supermarketId in: path description: ID of the supermarket containing the category to delete products from required: true schema: type: integer format: int64 - name: categoryId in: path description: ID of the category to delete products from required: true schema: type: integer format: int64	description: Supermarket not found
News	/news: post: tags: - news summary: Add a new news article. description: Add a new news article to the app operationId: addNews requestBody: description: The request body for creating a new news article. It should contain all necessary news details like title, image, publisher, intro, and fullText. content: application/json: schema: $ref: '#/components/schemas/News'	responses: '201': description: New news successfully added content: application/json: schema: $ref: '#/components/schemas/News' '400': description: Invalid input '422': description: Validation exception
	/news/{newsId}: put: tags: - news summary: Updates a specific news article. description: '' operationId: updateNewsWithForm parameters: - name: newsId in: path description: ID of the new that needs to be updated required: true	responses: '200': description: successful operation content: application/json: schema: $ref: '#/components/schemas/News' '400': description: Invalid ID supplied '404': description: News not found

	schema: type: integer format: int64 requestBody: description: The request body for updating an existing news article. It should contain all the fields that need to be updated for the specified news article (title, image, publisher, intro, or fullText). Fields that are not included in the request will remain unchanged. content: application/json: schema: $ref: '#/components/schemas/News'	
	/news: get: tags: - news summary: Find all news articles. description: Gets all news articles from the app operationId: getNews	responses: '200': description: Successful operation content: application/json: schema: type: array items: $ref: '#/components/schemas/News' '400': description: Invalid input '422': description: Validation exception
	/news/{newsId}: get: tags: - news summary: Find a specific news article by ID. description: Returns a single news article operationId: getNewsById parameters: - name: newsId in: path description: ID of news to get required: true schema: type: integer format: int64	responses: '200': description: successful operation content: application/json: schema: $ref: '#/components/schemas/News' '400': description: Invalid ID supplied '404': description: News not found
	/news/{newsId}: delete: tags: - news summary: Deletes a specific news article. description: Delete a specific new (by id). operationId: deleteNews	responses: '200': description: The specified new was successfully deleted '400': description: Invalid new's value '404':

	parameters: - name: newsId in: path description: Id of the new that needs to be deleted required: true schema: type: integer format: int64	description: New not found
Statistics	/statistics/positiveStats: post: tags: - statistics summary: Creates a new positive statistic. description: Creates a new positive statistic in the app. operationId: updateStatisticsPos requestBody: description: The request body for creating a new positive statistic. It should contain all necessary statistic details. content: application/json: schema: $ref: '#/components/schemas/Statistics'	responses: '201': description: New positive statistics successfully added content: application/json: schema: $ref: '#/components/schemas/Statistics' '400': description: Invalid ID supplied '404': description: Positive statistics not found '422': description: Validation exception
	/statistics/negativeStats: post: tags: - statistics summary: Creates a new negative statistic. description: Create new negative statistics in the app. operationId: updateStatisticsNeg requestBody: description: The request body for creating a new negative statistic. It should contain all necessary statistic details. content: application/json: schema: $ref: '#/components/schemas/Statistics'	responses: '201': description: New negative statistics successfully added content: application/json: schema: $ref: '#/components/schemas/Statistics' '400': description: Invalid ID supplied '404': description: Statistic not found '422': description: Validation exception
	/statistics/positiveStats: get: tags: - statistics summary: Finds all positive statistics. description: "" operationId: findStatisticsByStatusPos parameters:	responses: '200': description: successful operation content: application/json: schema: type: array

	- name: type in: query description: Status values that need to be considered for filter required: true explode: true schema: type: string default: positive	items: $ref: '#/components/schemas/Statistics' '400': description: Invalid status value
	/statistics/negativeStats: get: tags: - statistics summary: Finds all negative statistics. description: "" operationId: findStatisticsByStatusNeg parameters: - name: type in: query description: Status values that need to be considered for filter required: true explode: true schema: type: string default: negative	responses: '200': description: successful operation content: application/json: schema: type: array items: $ref: '#/components/schemas/Statistics' '400': description: Invalid status value
	/statistics/{statisticsId}: get: tags: - statistics summary: Find a specific statistic's details by Id. description: Returns a single statistics details operationId: getStatisticsById parameters: - name: statisticsId in: path description: ID of statistics to return required: true schema: type: integer format: int64	responses: '200': description: successful operation content: application/json: schema: $ref: '#/components/schemas/Statistics' '400': description: Invalid ID supplied '404': description: Statistic not found
	/statistics/{statisticsId}: put: tags: - statistics summary: Updates a specific statistic details. description: '' operationId: updatestatisticsWithForm parameters: - name: statisticsId	responses: '200': description: Statistic successfully updated content: application/json: schema: $ref: '#/components/schemas/Statistics'

	in: path description: ID of the statistic that needs to be updated required: true schema: type: integer format: int64 requestBody: description: The request body for updating an existing statistic. It should contain all the fields that need to be updated for the specified stat (numericInfo or/and type). Fields that are not included in the request will remain unchanged. content: application/json: schema: $ref: '#/components/schemas/Statistics'	'400': description: Invalid statistic's Id '404': description: Statistic not found
	/statistics/{statisticsId}: delete: tags: - statistics summary: Deletes a specific statistic. description: delete a specific statistic. operationId: deletetatistics parameters: - name: statisticsId in: path description: Statistics id to delete required: true schema: type: integer format: int64	responses: '200': description: Statistic successfully deleted '400': description: Invalid statistic's value '404': description: Statistic not found
Map	/maps: post: tags: - maps summary: Add a new map to the app. description: Add a new map to the app operationId: addMaps requestBody: description: The request body for creating a new map. It should contain all necessary map details like locationName, lat, and lon. content: application/json: schema: $ref: '#/components/schemas/Maps' required: true	responses: '201': description: New map successfully added content: application/json: schema: $ref: '#/components/schemas/Maps' '400': description: Invalid input '422': description: Validation exception
	/maps/{mapsId}: put:	responses: '200':

	tags: - maps summary: Updates a specific map. description: '' operationId: updateMapsWithForm parameters: - name: mapsId in: path description: ID of map that needs to be updated required: true schema: type: integer format: int64 requestBody: description: The request body for updating an existing map. It should contain all the fields that need to be updated for the specified map (locationName, lat, or lon). Fields that are not included in the request will remain unchanged. content: application/json: schema: $ref: '#/components/schemas/Maps' required: true	description: Map successfully updated content: application/json: schema: $ref: '#/components/schemas/Maps' '400': description: Invalid map ID supplied '404': description: Map not found
	/maps/{mapsId}: get: tags: - maps summary: Finds a specific map by Id. description: Returns a single map operationId: getMapsById parameters: - name: mapsId in: path description: ID of map to return required: true schema: type: integer format: int64	responses: '200': description: successful operation content: application/json: schema: $ref: '#/components/schemas/Maps' '400': description: Invalid ID supplied '404': description: Map not found
	/maps/{mapsId}: delete: tags: - maps summary: Deletes a specific map. description: deletes a map. operationId: deleteMaps parameters: - name: mapsId in: path description: Id of the map that needs to be deleted	responses: '200': description: A specific map was deleted. '400': description: Invalid maps's value. '404': description: Map not found.

	required: true schema: type: integer format: int64	