

# Wishify API Specification

<b>Endpoints - Basic.....</b>	<b>2</b>
Auth.....	2
Categories.....	2
Wishlists page.....	3
Events.....	5
Items.....	7
Users.....	8
Contributions.....	9
<b>Endpoints - in depth.....</b>	<b>11</b>
Authentication.....	11
POST /auth/register.....	11
POST /auth/login.....	12
POST /auth/logout.....	13
GET /auth/me.....	13
Wishlists.....	14
POST /wishlists.....	15
GET /wishlists/:id.....	16
DELETE /wishlists/:id.....	17
PUT /wishlists/:id.....	18
Users.....	19
GET /users.....	19
PUT /users.....	20
DELETE /users.....	21
GET /users/:id.....	22
Events.....	23
GET /events.....	23
POST /events.....	24
GET /events/:id.....	25
PUT /events/:id.....	26
DELETE /events/:id.....	28
Items.....	29
GET /items/:id.....	29
Categories.....	29

# Endpoints - Basic

## Auth

Request	Description	Input	Output
Post /register	Create a user account.	<b>email,</b> <b>password,</b> <b>displayName,</b> picture	email, password, displayName, picture
Post /login	Login to account	<b>email,</b> <b>password</b>	token
Post /logout	Logout of account	<b>token</b>	
Get /me	return user with ID.	<b>token</b>	email, password, displayName, picture
Post /forgot-password	Send an email with recovery code	<b>email</b>	200 if everything worked 400 otherwise
Post /reset-password	Reset password with one time code	<b>email,</b> <b>otc,</b> <b>newPassword</b>	200 if everything worked 400 otherwise

## Categories

Request	Description	Input	Output
Get /	Create a user account.		For[ id, name, description, created]
Get /:id	Login to account		id, name, description, created

## Wishlists page

Request	Description	Input	Output
Get /	Get all your wishlists.	<b>token</b>	For[ id, eventid. name, description, image, deadline, dateUpdated, dateCreated]
Post /	Make a wishlist	<b>token,</b> <b>name,</b> eventid, description, image,	wishlist_id 200 - success 400 - error otherwise
Get /id	Get a specific wishlist.	<b>token</b>	id, eventid. name, description, image, deadline, dateUpdated, dateCreated For items: [] For contributions []
Put /:id	Update wishlist details	<b>token,</b> <b>id,</b> (one or more of following): name, description, image, deadline,	id, eventid. name, description, image, deadline, dateUpdated, dateCreated 200 - success 400 - error otherwise
Delete /:id	Delete wishlist	<b>token</b>	200 - success 400 - error otherwise
Post /id/duplicate	Duplicate the wishlist	<b>token</b>	200 - success 400 - error otherwise
Delete /:id/members	remove a member	<b>token</b> <b>userId</b>	200 - success 400 - error otherwise

	(must be the owner to edit another users membership)		
Get /:id/members	get list of members	<b>token</b>	For members[ id, user_id, event_id, blind, owner, dateUpdated, dateCreated ]
Post /:id/members	add a membership (must be the owner)	<b>token</b> <b>userId</b> owner blind	200 - success 400 - error otherwise
Put /:id/members	Update a membership (must be the owner to edit another users membership)	<b>token</b> owner blind	id, user_id, event_id, blind, owner, dateUpdated, dateCreated
Get /:id/items	Get all items in a wishlist	<b>token</b>	For[ id, name, description, url, image, quantity, price, priority dateUpdated, dateCreated, user_displayName, user_id ]
GET /shared/:token	get the shared wishlist		id, eventid. name, description, image, deadline, dateUpdated, dateCreated For items: []

Post /share	Add user to wishlist with email	<b>wishlist_id,</b> <b>email</b>	200 success 400 otherwise
-------------	---------------------------------	-------------------------------------	------------------------------

## Events

Request	Description	Input	Output
Get /	Get all events.	<b>token</b>	For[ name, eventid. description, image, address, city, dateUpdated, dateCreated]
Post /	Make an event	<b>token,</b> <b>name,</b> description, image, address, city	200 - success 400 - error otherwise
Put /id	Make an event	<b>token,</b> <b>name,</b> description, image, address, city	200 - success 400 - error otherwise
Delete /id	Delete event	<b>token</b>	200 - success 400 - error otherwise
Get /:id/wishlists	get list of wishlists under an event	<b>token</b>	For wishlist[ id, eventid. name, description, image, dateUpdated, dateCreated]
Get /:id/members	get list of members under an event	<b>token</b>	For members[ id, user_id, event_id, owner, dateUpdated,

			dateCreated ]
Post /:id/members	add a member to an event (must be the owner to edit another users membership)	<b>token</b> <b>userId</b> owner	200 - success 400 - error otherwise
Delete /:id/members	remove a member from an event (must be the owner to edit another users membership)	<b>token</b> <b>userId</b>	200 - success 400 - error otherwise
Put /:id/members	Update a membership (must be the owner to edit another users membership)	<b>token</b> owner	id, user_id, event_id, owner, dateUpdated, dateCreated

## Items

Request	Description	Input	Output
Get /:id	Get item details	<b>token</b>	wishlists_id name, quantity, description, url, image, price
PUT /:id	Update an item	<b>Token</b> name, description, url, image, quantity, price	wishlists_id name, quantity, description, url, image, price
PUT /	Update a array of items	For[ <b>Token</b> name, description, url, image, quantity, price]	For[ <b>Token</b> name, description, url, image, quantity, price]
DELETE /:id	Remove an item	<b>token</b>	200 - success 400 - error otherwise
POST /	create an item	<b>token</b> <b>wishlists_id</b> <b>name,</b> <b>quantity,</b> description, url, image, price	wishlists_id name, quantity, description, url, image, price

## Users

Request	Description	Input	Output
Get /	return current user profile and categories	<b>token</b>	user: id email, displayName, bio, picture, datecreated, dateupdated For categories:[ id name description created love]
Put /	edit your account	<b>token,</b> email, displayName, bio, picture, password, newPassword	Updated input data  400 error if new email already in DB
Delete /	delete your account (must include password)	<b>token,</b> <b>password</b>	200 - success 400 error otherwise
Get /userId	return user with ID and categories	<b>token</b>	user: displayName, email bio, picture For categories:[ id name description love]
Post /categories/:id	Add category to user	<b>token</b> <b>love</b>	200 - success 400 error otherwise
Delete /categories/:id	Remove category from user	<b>token</b>	200 - success 400 error otherwise
Put /categories/:id	Update a category love	<b>token</b> <b>love</b>	200 - success 400 error otherwise



## Contributions

Request	Description	Input	Output
Get /	Get all contributions from logged in user.	<b>token</b>	For[ id, item_id, quantity, purchased, note, dateUpdated, dateCreated, user_displayName, user_id]
Get /contributions/wishlists/:id	Get all contributions from wishlist	<b>token</b>	For[ id, item_id, quantity, purchased, note, dateUpdated, dateCreated, user_displayName, user_id]
GET /contributions/items/:id	Get contributions for an item	<b>token</b>	For[ id, item_id, quantity, purchased, note, dateUpdated, dateCreated, user_displayName, user_id]
POST /contributions	Add a contribution (given item_id)	<b>token</b> <b>item_id</b> , quantity, purchased, note	id, item_id, quantity, purchased, note, dateUpdated, dateCreated
PUT /contributions/:id	Update a contribution (mark as purchased, etc.)	<b>token</b> quantity, purchased, note	id, item_id, quantity, purchased, note,

			dateUpdated, dateCreated, user_displayName, user_id
DELETE /contributions/:id	Remove a contribution	<b>token</b>	200 - success 400 error otherwise

# Endpoints - in depth

- /auth/login accepts a password and returns a JWT token.
- The rest of the endpoints require an authorization header containing a valid JWT token (excluding /auth/register).
  - E.g. "Authorization": "Bearer <JWT Token>"
- **Bolded** fields are **required**

## Authentication

### POST /auth/register

- Used to register a user

#### Request Body

```
{
  email,
  password,
  displayName,
  picture
}
```

#### Response 201 Created

##### Response Body

```
{
  "message": "User registered successfully",
  "user": {
    "id": <id>,
    "password": "<password>"
    "displayname": "<displayName>"
  }
}
```

#### Response 400 Bad Request

Either email, password, or displayName is missing

```
{
  "message": "email, password and displayName are required"
}
```

#### Response 409 Conflict

Email already registered in database

```
{
  "message": "Email is already in use"
}
```

Response **500** Internal Server Error

- Something went wrong.

## POST /auth/login

Used to receive a JWT token for authentication

- Lasts 24h

Request Body

```
{
  email,
  password
}
```

Response **200** OK

Response Body

```
{
  "message": "Login successful",
  "token": "<token>"
}
```

Response **400** Bad Request

If either email or password is missing

```
{
  "message": "email and password are required"
}
```

Response **401** Unauthorized

Wrong password or email

```
{
  "message": "Invalid email"
}
```

```
{
  "message": "Invalid password"
}
```

Response **500** Internal Server Error

- Something went wrong.

## POST /auth/logout

Used to log user out and invalidate token

Request Header

```
"Authorization": "Bearer <JWT Token>"
```

Response **200** OK

Response Body

```
{  
  "message": "Logout successful"  
}
```

Response **401** Unauthorized

No token provided

```
{  
  "message": "No token provided"  
}
```

Response **500** Internal Server Error

- Something went wrong.

## GET /auth/me

Used to get information about a user. Returns information based on a valid JWT token.

Request Header

```
"Authorization": "Bearer <JWT Token>"
```

Response **200** OK

Response Body

```
{  
  "id": <id>,  
  "displayname": "<displayName>",  
  "email": "<email>",  
  "picture": "/assets/<img>"  
}
```

```
}
```

Response **401** Unauthorized

No token provided

```
{  
  "message": "No token provided"  
}
```

Invalid token

```
{  
  "message": "Invalid token"  
}
```

Response **404** Not Found

User was removed but token still active

```
{  
  "message": "User not found"  
}
```

Response **500** Internal Server Error

- Something went wrong.

## Wishlists

GET /wishlists

Get list of wishlists that the user is a member of (or is a member of an event wishlist)

Response **200** OK

Returns an array of the users wishlists.

Response Body

```
[  
  {  
    "id": <wishlist_id>,  
    "name": "<wishlist_name>",  
    "description": <description>,  
    "image": <image>,  
    "dateupdated": "<dateupdated>",  
    "datecreated": "<datecreated>"  
  },  
  {...}  
]
```

```
] ]
```

Response **401** Unauthorized

No token provided

```
{
  "message": "Access denied. No token provided."
}
```

Response **403** Forbidden

Invalid token

```
{
  "message": "Invalid token."
}
```

POST /wishlists

Request Header

```
"Authorization": "Bearer <JWT Token>"
```

Request Body

```
{
  name,
  eventid,
  description,
  image
}
```

Response **201** Created

Response Body

```
{
  "wishlist_id": <wishlist_id>,
  "message": "Wishlist created successfully!"
}
```

Response **400** Bad Request

name is missing

```
{
  "error": "name is required"
}
```

```
}
```

Response **500** Internal Server Error

- Something went wrong.

GET /wishlists/:id

“:id” is the parameter for the wishlist id.

Get information of a single wishlist

Request Header

```
“Authorization”: “Bearer <JWT Token>”
```

Response **200** OK

Response Body

```
{
  "id": <id>,
  "name": "<name>",
  "description": "<description>",
  "image": "<image>",
  "dateupdated": "<dateupdated>",
  "datecreated": "<datecreated>",
  "event_id": "<event_id>"
}
```

Response **401** Unauthorized

No token provided

```
{
  "message": "Access denied. No token provided."
}
```

Response **403** Forbidden

Invalid token

```
{
  "message": "Invalid token."
}
```

Response **404** Not Found

wishlist does not exist

```
{
```



```
{
  "error": "wishlist not found."
}
```

Response **500** Internal Server Error

- Something went wrong.

## DELETE /wishlists/:id

“:id” is the parameter for the wishlist id.

Delete a single wishlist you are the owner of.

Request Header

```
“Authorization”: “Bearer <JWT Token>”
```

Response **200** OK

Response Body

```
{
  "message": "Wishlist deleted successfully."
}
```

Response **401** Unauthorized

No token provided

```
{
  "message": "Access denied. No token provided."
}
```

Response **403** Forbidden

Invalid token

```
{
  "message": "Invalid token."
}
```

You are not a member of the wishlist

```
{
  "error": "Access denied. You are not a member of this wishlist."
}
```

You are not an owner of the wishlist

```
{
  "error": "Only the owner can delete this wishlist."
}
```

Response **500** Internal Server Error

- Something went wrong.

## PUT /wishlists/:id

“:id” is the parameter for the wishlist id.

Edit a single wishlist you are the owner of.

### Request Header

```
“Authorization”: “Bearer <JWT Token>”
```

### Request Body

All are optional. You only have to include what you are changing

```
{
  name,
  eventid,
  description,
  image
}
```

Response **200** OK

### Response Body

Response contains the updated data

```
{
  "message": "Wishlist updated successfully.",
  "wishlist": {
    "id": "<id>",
    "event_id": "<event_id>",
    "name": "<wishlist_name>",
    "description": "<description>",
    "image": "<image>",
    "dateupdated": "<dateupdated>",
    "datecreated": "<datecreated>"
  }
}
```

Response **401** Unauthorized

No token provided

```
{
  "message": "Access denied. No token provided."
}
```

Response **403** Forbidden

Invalid token

```
{
  "message": "Invalid token."
}
```

You are not a member of the wishlist

```
{
  "error": "Access denied. You are not a member of this wishlist."
}
```

You are not an owner of the wishlist

```
{
  "error": "Only the owner can edit this wishlist."
}
```

Response **404** Not Found

wishlist does not exist

```
{
  "error": "wishlist not found."
}
```

Response **500** Internal Server Error

- Something went wrong.

## Users

GET /users

Returns information for the logged in user based on the provided token.

Response **200** OK

Response Body

```
{
  "id": <id>,
  "email": "<email>",
}
```

```
"displayname": "<displayName>",  
"picture": "<link to image>",  
"datecreated": "<datecreated>",  
"dateupdated": "<dateupdated>"  
}
```

Response **404** Not Found

User not found

```
{  
  "message": "User not found"  
}
```

Response **500** Internal Server Error

- Something went wrong.

## PUT /users

Returns information for the logged in user based on the provided token.

Request Header

```
"Authorization": "Bearer <JWT Token>"
```

Request Body

If newPassword is provided, then the current password is required.

```
{  
  email,  
  displayName,  
  picture,  
  password,  
  newPassword  
}
```

Response **200** OK

Response Body

```
{  
  "message": "Profile updated",  
  "user": {  
    "id": <id>,  
    "email": "<email>",  
    "displayname": "<displayName>",  
    "picture": "<link to image>",  
    "datecreated": "<datecreated>",  
  }  
}
```

```
    "dateupdated": "<dateupdated>"
  }
}
```

#### Response 400 Bad Request

newPassword is provided but password is not.

```
{
  "message": "Current password is required to change password"
}
```

#### Response 403 Forbidden

Incorrect password when attempting to change the user's password

```
{
  "message": "Incorrect password"
}
```

#### Response 404 Not Found

User not found

```
{
  "message": "user not found"
}
```

#### Response 500 Internal Server Error

- Something went wrong.

## DELETE /users

Delete the logged-in user's account

#### Request Header

```
"Authorization": "Bearer <JWT Token>"
```

#### Request Body

```
{
  password
}
```

#### Response 200 OK

#### Response Body

```
{
  "message": "Account deleted successfully"
}
```

Response 400 Bad Request

Password is required.

```
{
  "message": "Password is required to delete account"
}
```

Response 403 Forbidden

```
{
  "message": "Incorrect password"
}
```

Response 404 Not Found

User not found

```
{
  "message": "user not found"
}
```

Response 500 Internal Server Error

- Something went wrong.

GET /users/:id

Response 200 OK

Response Body

```
{
  "id": <id>,
  "displayname": "<displayName>",
  "picture": "<link to picture>",
  "datecreated": "<dateCreated>"
}
```

Response 404 Not Found

User not found

```
{
  "message": "User not found"
}
```

Response **500** Internal Server Error

- Something went wrong.

## Events

### GET /events

Returns all the events the user is a part of

Request Header

```
"Authorization": "Bearer <JWT Token>"
```

Response **200** OK

Response Body

```
[
  {
    "id": "<event_id>",
    "name": "<event_name>",
    "description": "<description>",
    "url": "<url>",
    "addr": "<address>",
    "city": "<city>",
    "image": "<image>",
    "dateupdated": "<dateupdated>",
    "datecreated": "<datecreated>"
  },
  {...}
]
```

Response **401** Unauthorized

No token provided

```
{
  "message": "Access denied. No token provided."
}
```

Response **403** Forbidden

Invalid token

```
{
```

```
{
  "message": "Invalid token."
}
```

Response **500** Internal Server Error

- Something went wrong.

## POST /events

Create an event.

Request Header

```
"Authorization": "Bearer <JWT Token>"
```

Request Body

```
{
  name,
  description,
  image,
  address,
  city
}
```

Response **201** Created

Response body

```
{
  "event_id": 1,
  "message": "Event created successfully!"
}
```

Response **400** Bad Request

name is missing

```
{
  "error": "name is required"
}
```

Response **401** Unauthorized

No token provided

```
{
```



```
{
  "message": "Access denied. No token provided."
}
```

Response **403** Forbidden

Invalid token

```
{
  "message": "Invalid token."
}
```

Response **500** Internal Server Error

- Something went wrong.

## GET /events/:id

Retrieve information on specific event. The user must be a member of the event.

Request Header

```
"Authorization": "Bearer <JWT Token>"
```

Response **200** OK

Response body

```
{
  "id": "<event_id>",
  "name": "<event_name>",
  "description": "<description>",
  "url": "<url>",
  "addr": "<address>",
  "city": "<city>",
  "image": "<image>",
  "dateupdated": "<dateupdated>",
  "datecreated": "<datecreated>"
}
```

Response **401** Unauthorized

No token provided

```
{
  "message": "Access denied. No token provided."
}
```

## Response 403 Forbidden

Invalid token

```
{
  "message": "Invalid token."
}
```

## Response 404 Not Found

Event not found

```
{
  "error": "event not found."
}
```

## Response 500 Internal Server Error

- Something went wrong.

## PUT /events/:id

Update the information for an event.

You must be the owner to update the event.

Request Header

```
"Authorization": "Bearer <JWT Token>"
```

Request Body

All fields are optional. You only have to include what you are changing.

```
{
  name,
  description,
  image,
  address,
  city
}
```

## Response 200 OK

Response body

```
{
  "message": "Event updated successfully.",
  "event": {
    "id": "<event_id>",
    "name": "<event_name>",
    "description": "<description>",
  }
}
```

```
"url": "<url>",
"addr": "<address>",
"city": "<city>",
"image": "<image>",
"dateupdated": "<dateupdated>",
"datecreated": "<datecreated>"
}
}
```

Response **401** Unauthorized

No token provided

```
{
  "message": "Access denied. No token provided."
}
```

Response **403** Forbidden

Invalid token

```
{
  "message": "Invalid token."
}
```

You are not a member of the event

```
{
  "error": "Access denied. You are not a member of this event."
}
```

You are not an owner of the event

```
{
  "error": "Only the owner can edit this event."
}
```

Response **404** Not Found

wishlist does not exist

```
{
  "error": "Event not found."
}
```

Response **500** Internal Server Error

- Something went wrong.

## DELETE /events/:id

Delete a specific event.

You must be the owner to delete the event.

### Request Header

```
"Authorization": "Bearer <JWT Token>"
```

### Response 200 OK

#### Response body

```
{
  "message": "event deleted successfully."
}
```

### Response 401 Unauthorized

#### No token provided

```
{
  "message": "Access denied. No token provided."
}
```

### Response 403 Forbidden

#### Invalid token

```
{
  "message": "Invalid token."
}
```

#### You are not a member of the event

```
{
  "error": "Access denied. You are not a member of this event."
}
```

#### You are not an owner of the event

```
{
  "error": "Only the owner can delete this event."
}
```

### Response 500 Internal Server Error

- Something went wrong.

# Items

## GET /items/:id

Retrieve information on specific item.

Request Header

```
"Authorization": "Bearer <JWT Token>"
```

Response 200 OK

Response body

```
{  
  "To be completed..."  
}
```

Response 401 Unauthorized

No token provided

```
{  
  "message": "Access denied. No token provided."  
}
```

Response 403 Forbidden

Invalid token

```
{  
  "message": "Invalid token."  
}
```

Response 404 Not Found

Event not found

```
{  
  "error": "Item not found."  
}
```

Response 500 Internal Server Error

- Something went wrong.

## Categories