

Wishify API Specification

Endpoints - Basic.....	2
Auth.....	2
Categories.....	2
Wishlists page.....	3
Events.....	5
Items.....	7
Users.....	8
Contributions.....	9
Ideas.....	10
Endpoints - in depth.....	11
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.	email, password, displayName	email, password, displayName, picture
Post /login	Login to account	email, password	token
Post /logout	Logout of account	token	
Get /me	return user with ID.	token	email, password, displayName, picture
Post /forgot-password	Send an email with recovery code	email	200 if everything worked 400 otherwise
Post /reset-password	Reset password with one time code	email, otc, newPassword	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.	token	For[id, eventid. name, description, image, deadline, dateUpdated, dateCreated]
Post /	Make a wishlist	token, name, eventid, description, image,	wishlist_id 200 - success 400 - error otherwise
Get /id	Get a specific wishlist.	token	id, eventid. name, description, image, deadline, dateUpdated, dateCreated For items: [] For contributions []
Put /:id	Update wishlist details	token, id, (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	token	200 - success 400 - error otherwise
Post /id/duplicate	Duplicate the wishlist	token	200 - success 400 - error otherwise
Delete /:id/members	remove a member	token userId	200 - success 400 - error otherwise

	(must be the owner to edit another users membership)		
Get /:id/members	get list of members	token	For members[id, user_id, event_id, blind, owner, dateUpdated, dateCreated]
Post /:id/members	add a membership (must be the owner)	token userId owner blind	200 - success 400 - error otherwise
Put /:id/members	Update a membership (must be the owner to edit another users membership)	token owner blind	id, user_id, event_id, blind, owner, dateUpdated, dateCreated
Get /:id/items	Get all items in a wishlist	token	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	wishlist_id, email	200 success 400 otherwise
-------------	---------------------------------	-------------------------------------	------------------------------

Events

Request	Description	Input	Output
Get /	Get all events.	token	For[name, eventid. description, image, address, city, dateUpdated, dateCreated]
Post /	Make an event	token, name, description, image, address, city	200 - success 400 - error otherwise
Put /id	Make an event	token, name, description, image, address, city	200 - success 400 - error otherwise
Delete /id	Delete event	token	200 - success 400 - error otherwise
Get /:id/wishlists	get list of wishlists under an event	token	For wishlist[id, eventid. name, description, image, dateUpdated, dateCreated]
Get /:id/members	get list of members under an event	token	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)	token userId owner	200 - success 400 - error otherwise
Delete /:id/members	remove a member from an event (must be the owner to edit another users membership)	token userId	200 - success 400 - error otherwise
Put /:id/members	Update a membership (must be the owner to edit another users membership)	token owner	id, user_id, event_id, owner, dateUpdated, dateCreated

Items

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

Users

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

POST /upload	upload a profile picture	token Formdata file: 'picture'	200 - success 400 error otherwise
--------------	--------------------------	---	--------------------------------------

Contributions

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

PUT /contributions/:id	Update a contribution (mark as purchased, etc.)	token quantity, purchased, note	id, item_id, quantity, purchased, note, dateUpdated, dateCreated, user_displayName, user_id
DELETE /contributions/:id	Remove a contribution	token	200 - success 400 error otherwise

Ideas

Request	Description	Input	Output
Get /	return current user profile and categories	token	id, name, description url, image, price, sponsored uses, created, categories [{ Id, name}] score
Get /trending	Get the top 10 most popular items.	(No input required)	id, name, description url, image, price, sponsored uses, created, categories [{ Id, name}]

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