Food Celebrator API Documentation

Authentication Endpoints

- `POST /api/auth/register` → Register a new user
- `POST /api/auth/login` → Log in a user and return a token
- `GET /api/auth/me` → Get logged-in user data (requires authentication)

User Endpoints

- `POST /api/users` → Create a new user
- `GET /api/users` → Fetch all users
- `DELETE /api/users/:id` → Delete a user by ID

Picture Endpoints

- `POST /api/pictures` → Create a new picture
- `GET /api/pictures` → Fetch all pictures
- `GET /api/pictures/:pictureId` → Fetch a picture by ID
- `DELETE /api/pictures/:id` → Delete a picture by ID

User-Picture Linking Endpoints

- `POST /api/users/:userId/pictures/:pictureId` → Link a user to a picture
- `GET /api/users/:userId/pictures` → Fetch all pictures linked to a user
- `DELETE /api/users/:userId/pictures/:pictureId` → Unlink a user from a picture
- `GET /api/users_x_pictures` → Fetch all user-picture links

Username-based Picture Endpoints

- `GET /api/username/:username/pictures` → Fetch all pictures uploaded by a specific username
- `GET /api/username/:username/pictures/:pictureid` → Fetch a specific picture by a username and picture ID

Feed Endpoints

- `GET /api/feed` → Fetch the global picture feed
- `GET /api/feed?limit=:limit&offset=:offset` → Fetch paginated picture feed

Image Upload Endpoint

- `POST /api/upload` → Upload an image (performs face detection and object classification before uploading)

Comment Endpoints

- `POST /api/createComment` → Create a comment on a picture
- `PUT /api/editComment` → Edit a comment
- `DELETE /api/deleteComment` → Delete a comment
- `GET /api/:pictureId/comments` → Fetch all comments for a specific picture

Like Endpoints

- `POST /api/createLike` → Create a like on a picture
- `DELETE /api/deleteLike` → Delete a like
- `GET /api/:pictureId/likes` → Fetch all likes for a specific picture

Bio and Profile Picture Endpoints

- `PUT /api/:username/bio` → Update user bio by username
- `GET /api/:username/bio` → Fetch user bio by username
- `GET /api/:username/profilepic` → Fetch user profile picture number by username
- `PUT /api/:username/profilepic` → Update user profile picture number by username

Static File Serving

- `GET /` → Serve the frontend index page
- `GET /assets/*` → Serve static assets from the frontend build

Further Explanation

POST /api/auth/register

- **Description:** Registers a new user.

- **Request Body:** JSON object containing:
  - `username` (string)
  - `password` (string)
  - `email` (string)
- **Response:** Returns the newly created user data.
- **Errors:** May return error details if the input is invalid or the username/email is already taken.

POST /api/auth/login

- **Description:** Authenticates a user and returns a token.
- **Request Body:** JSON object containing:
  - `username` (string)
  - `password` (string)
- **Response:** JSON object with a `token` field.
- **Errors:** Returns error if the username is not found or the password is incorrect.

GET /api/auth/me

- **Description:** Retrieves data for the logged-in user.
- **Headers:** Must include an `Authorization` header with a valid token.
- **Response:** User data (without password).
- **Errors:** Returns an error if authentication fails.

---

2. User Endpoints

POST /api/users

- **Description:** Creates a new user.
- **Request Body:** JSON object containing:
  - `username` (string)
  - `password` (string)
  - `email` (string)

- **Response:** Returns the created user object.
- **Errors:** Similar to registration errors.

GET /api/users

- **Description:** Fetches a list of all users.
- **Response:** JSON array of users (each with `id`, `username`, `email`, and `created_at`).
- **Errors:** Returns an error on failure.

DELETE /api/users/:id

- **Description:** Deletes a user by their unique ID.
- **Path Parameter:**
  - `id` (UUID of the user)
- **Response:** Returns the deleted user record.
- **Errors:** Returns error if the user is not found.

---

3. Picture Endpoints

POST /api/pictures

- **Description:** Creates a new picture record.
- **Request Body:** JSON object containing:
  - `URL` (string): The URL of the image.
  - `caption` (string): An optional caption.
- **Response:** Returns the newly created picture object.
- **Errors:** Returns error details if creation fails.

GET /api/pictures

- **Description:** Retrieves all pictures.
- **Response:** JSON array of pictures (with fields like `id`, `URL`, `caption`, and `created_at`).

GET /api/pictures/:pictureId

- **Description:** Retrieves a picture by its ID.
- **Path Parameter:**
  - `pictureId` (UUID)
- **Response:** JSON object of the requested picture.
- **Errors:** Returns error if the picture is not found.

DELETE /api/pictures/:id

- **Description:** Deletes a picture by its unique ID.
- **Path Parameter:**
  - `id` (UUID)
- **Response:** Returns the deleted picture record.
- **Errors:** Returns error if deletion fails.

---

4. User-Picture Linking Endpoints

POST /api/users/:userId/pictures/:pictureId

- **Description:** Links a user to a picture (creates an association).
- **Path Parameters:**
  - `userId` (UUID)
  - `pictureId` (UUID)
- **Response:** Returns the created user-picture link record.
- **Errors:** Returns error if linking fails.

GET /api/users/:userId/pictures

- **Description:** Fetches all pictures linked to a specific user.
- **Path Parameter:**
  - `userId` (UUID)

- **Response:** JSON array of pictures associated with the user.
- **Errors:** Returns error if the user is not found.

DELETE /api/users/:userId/pictures/:pictureId

- **Description:** Removes the link between a user and a picture.
- **Path Parameters:**
  - `userId` (UUID)
  - `pictureId` (UUID)
- **Response:** Returns the deleted link record.
- **Errors:** Returns error if the link does not exist.

GET /api/users_x_pictures

- **Description:** Fetches all user-picture link records.
- **Response:** JSON array of all links between users and pictures.
- **Errors:** Returns error if retrieval fails.

---

5. Username-based Picture Endpoints

GET /api/username/:username/pictures

- **Description:** Fetches all pictures uploaded by a specific username.
- **Path Parameter:**
  - `username` (string)
- **Response:** JSON array containing picture data associated with the username.
- **Errors:** Returns error if the user is not found.

GET /api/username/:username/pictures/:pictureid

- **Description:** Fetches a specific picture by a username and picture ID.
- **Path Parameters:**

- `username` (string)
- `pictureid` (UUID)

- **Response:** JSON object with the picture details.
- **Errors:** Returns error if no matching picture is found.

---

6. Feed Endpoints

GET /api/feed

- **Description:** Retrieves the global picture feed.
- **Response:** JSON array of feed items (includes picture URL, caption, creation time, associated username, and profile picture number).

GET /api/feed?limit=:limit&offset=:offset

- **Description:** Retrieves a paginated picture feed.
- **Query Parameters:**
  - `limit` (number): The maximum number of items to return.
  - `offset` (number): The number of items to skip.
- **Response:** JSON array of feed items.
- **Errors:** Returns error if `limit` or `offset` are invalid.

---

7. Image Upload Endpoint

POST /api/upload

- **Description:** Uploads an image with several validation steps:
  - **Face Detection:** Rejects the image if a face is detected.
  - **Object Localization:** Checks for person objects and rejects if found.
  - **Label Detection:** Validates labels against allowed labels.
- **Request:**
  - **Content Type:** `multipart/form-data`
  - **Fields:**
    - `image` : The image file.

    - `caption`: (Optional) A caption for the picture.
- **Headers:** Must include a valid `Authorization` token.
- **Process:**
    1. Validates presence of an image and token.
    2. Uses Google Cloud Vision API for image analysis.
    3. Saves the image temporarily and uploads it to an external server.
    4. Creates a picture record and links it to the user.
- **Response:** JSON object containing a success message, the new picture record, and logs of the processing steps.
- **Errors:** Returns error if image validations fail or if the upload encounters issues.

---

8. Comment Endpoints

POST /api/createComment

- **Description:** Creates a comment on a picture.
- **Request Body:** JSON object containing:
    - `user_id` (UUID)
    - `picture_id` (UUID)
    - `content` (string)
- **Response:** JSON object with a success message and the created comment.
- **Errors:** Returns error if comment creation fails.

PUT /api/editComment

- **Description:** Edits an existing comment.
- **Request Body:** JSON object containing:
    - `comment_id` (UUID)
    - `content` (string): The updated comment text.
- **Response:** JSON object with the updated comment record.
- **Errors:** Returns error if the comment is not found or the update fails.

DELETE /api/deleteComment

- **Description:** Deletes a comment.
- **Request Body:** JSON object containing:
  - `comment_id` (UUID)
- **Response:** Returns the deleted comment record.
- **Errors:** Returns error if deletion fails.

GET /api/:pictureId/comments

- **Description:** Fetches all comments for a specific picture.
- **Path Parameter:**
  - `pictureId` (UUID)
- **Response:** JSON array of comment records including:
  - User details (`user_id` , `username`)
  - Comment content and creation time.
- **Errors:** Returns error if the picture or comments are not found.

---

9. Like Endpoints

POST /api/createLike

- **Description:** Creates a like on a picture.
- **Request Body:** JSON object containing:
  - `user_id` (UUID)
  - `picture_id` (UUID)
- **Response:** JSON object with a success message and the created like record.
- **Errors:** Returns error if the like cannot be created.

DELETE /api/deleteLike

- **Description:** Deletes a like.
- **Request Body:** JSON object containing:
  - `like_id` (UUID)
- **Response:** Returns the deleted like record.

- **Errors:** Returns error if deletion fails.

GET /api/:pictureId/likes

- **Description:** Fetches all likes for a specific picture.
- **Path Parameter:**
  - `pictureId` (UUID)
- **Response:** JSON array containing details of each like including:
  - User information (`user_id`, `username`)
  - The like's ID and creation time.
- **Errors:** Returns error if retrieval fails.

---

10. Bio and Profile Picture Endpoints

PUT /api/:username/bio

- **Description:** Updates a user's bio.
- **Path Parameter:**
  - `username` (string)
- **Request Body:** JSON object containing:
  - `updatedBio` (string)
- **Response:** Returns the updated bio.
- **Errors:** Returns error if the user is not found.

GET /api/:username/bio

- **Description:** Retrieves a user's bio.
- **Path Parameter:**
  - `username` (string)
- **Response:** JSON object containing the bio.
- **Errors:** Returns error if the user is not found.

GET /api/:username/profilepic

- **Description:** Retrieves the profile picture number for a user.
- **Path Parameter:**
  - `username` (string)
- **Response:** JSON object containing:
  - `profile_pic_num` (number)
- **Errors:** Returns error if the user is not found.

PUT /api/:username/profilepic

- **Description:** Updates the profile picture number for a user.
- **Path Parameter:**
  - `username` (string)
- **Request Body:** JSON object containing:
  - `profile_pic_num` (number)
- **Response:** Returns the updated profile picture number.
- **Errors:** Returns error if the user is not found.

---

11. Static File Serving

GET /

- **Description:** Serves the frontend index page.
- **Response:** HTML page (the client-side application).

GET /assets/*

- **Description:** Serves static assets (e.g., JavaScript, CSS, images) from the frontend build.
- **Response:** Returns the requested static file.
