Lab Exercise #6: Appendix - REST endpoint documentation

Endpoint[1]	**GET**[2]	**POST**[3]	**PUT**[4]	**DELETE**[5]
/api/admin/users	Description: Retrieves a paginated list of users with optional search and role filtering. Query Params: page (number), pageSize (number), search (string), role ("admin", "manager", "creator", "explorer") Success: 200 OK Returns: { items: UserDTO[], total: number, page:	-	-	-

[1] Endpoint of the resource and/or parameters
[2] GET method description and returned HTTP status code with the description.
[3] POST method description and returned HTTP status code with the description.
[4] PUT method description and returned HTTP status code with the description.
[5] DELETE method description and returned HTTP status code with the description.

	number, pageSize: number }			
/api/admin/users/:id	Description: Retrieves the detailed profile of a specific user by ID. URL Params: id (numeric user ID) Success: 200 OK Returns: UserDTO { id, name, email, role, createdAt }	-	-	-
/api/admin/users/:id	-	-	PATCH method Description: Updates a user's role. Triggers a ROLE_UPDATE audit log. URL Params: id (numeric user ID) Request Body: { "role": "admin" | "manager" | "creator" | "explorer" }	-

			Success: 200 OK Returns: Updated UserDTO	
/api/admin/users/:id/reset-password	-	Description: Forces a password reset to a system default (or env variable). Triggers a PASSWORD_RESET audit log. URL Params: id (numeric user ID) Success: 204 No Content Returns: None	**-**	**-**
/api/recipes/:recipeid/comments	Description: Fetches all comments for a specific recipe, including the author's username. Returns: Array of comment objects. Status: 200 OK.	Description: Creates a new comment for a recipe. Expects: { body: string } or { content: string }. Returns: The created comment object. Status: 201 Created, 400 Bad Request (empty	**-**	**-**

		content).		
/api/recipes/:recipeid/ratings/summary	Description: Gets the average rating and total count for a recipe. If logged in, includes the current user's rating. Returns: Summary object (avgRating, count, myRating). Status: 200 OK, 404 Not Found (recipe missing).	-	**-**	**-**
/api/recipes/:recipeid/ratings	-	Description: Upserts (creates or updates) a user's rating for a recipe. Expects: { value: number } (1-5). Returns: Updated summary object. Status: 200 OK, 400 Bad Request (invalid value), 404 Not Found.	**-**	**-**
/api/favorites	Description: Returns a paginated list of	-	**-**	**-**

	the user's favorited recipes, optionally filtered by text search. Expects: Query params: page, pageSize, textSearch. Returns: Object containing items array and metadata. Status: 200 OK.			
/api/favorites/:recipeId	Description: Checks if a specific recipe is in the user's favorites. Returns: Boolean (true/false). Status: 200 OK	Description: Toggles the favorite status (adds if missing, removes if exists). Returns: { isFavorite: boolean }. Status: 200 OK.	**-**	**-**
/api/reports	-	Description: Files a report against a recipe or comment. Expects: { targetType, targetId, reason, details }.	**-**	**-**

		Returns: Formatted report object. Status: 201 Created, 400 Bad Request (invalid type/params), 404 Not Found.		
/api/management/moderation	Description: (Admins/Mods only) Paginated list of all reports, filterable by status or type. Expects: Query params: status, type. Returns: Object with items array and metadata. Status: 200 OK.	-	-	-
/api/management/moderation/:reportId/resolve	-	Description: (Admins/Mods only) Marks a report as resolved without taking destructive action. Returns: Updated report object.	-	-

		Status: 200 OK, 404 Not Found.		
/api/management/moderation/:reportId/remove	-	Description: (Admins/Mods only) Marks a report as resolved without taking destructive action. Returns: Updated report object. Status: 200 OK, 404 Not Found.	-	-
/api/comments/:id	Description: Fetches details for a single comment by ID. Returns: Comment object with username. Status: 200 OK, 404 Not Found.	-	Description: Updates the text of an existing comment. Restricted to author or moderators. Expects: { content: string }. Returns: Updated comment object. Status: 200 OK, 403 Forbidden (not owner), 404 Not Found.	Description: Deletes a comment. Allowed for author, recipe owner, or moderators. Returns: Empty body. Status: 204 No Content, 403 Forbidden, 404 Not Found.
/api/ratings/:id	Description: Fetches details for a single rating by ID.	-	Description: Updates a rating's numeric value. Restricted to	Description: Deletes a rating. Restricted to author or moderators.

	Returns: Rating object with username. Status: 200 OK, 404 Not Found		author or moderators. Expects: { rating_value: number }. Returns: Updated rating object. Status: 200 OK, 400 Bad Request, 403 Forbidden.	Returns: Empty body. Status: 204 No Content, 403 Forbidden, 404 Not Found.
/api/categories	Description: Retrieves a list of all recipe categories. Returns: Array of category objects { category_id, name, description }. Status: 200 OK.	Description: (Admin/Mod only) Creates a new recipe category and logs the action. Expects: { name: string, description: string }. Returns: The created category object. Status: 201 Created, 400 Bad Request (duplicate name or missing fields).	**-**	**-**
/api/categories/:id	Description: Fetches details for a specific category by its ID.	-	Description: (Admin/Mod only) Updates the name and description of an existing category.	Description: (Admin/Mod only) Removes a category from the database.

	Returns: A single category object. Status: 200 OK, 404 Not Found.		Expects: { name: string, description: string }. Returns: The updated category object. Status: 200 OK, 404 Not Found.	Returns: Empty body. Status: 204 No Content, 404 Not Found.
/api/tags	Description: Retrieves all available recipe tags (e.g., "Vegan", "Spicy"). Returns: Array of tag objects. Status: 200 OK	Description: (Admin/Mod only) Adds a new tag to the system. Expects: { tag_name: string }. Returns: The created tag object. Status: 201 Created.	**-**	**-**
/api/tags/:id	-	-	Description: (Admin/Mod only) Renames an existing tag. Expects: { tag_name: string }.	Description: (Admin/Mod only) Deletes a tag by its ID. Returns: Empty body. Status: 204 No Content, 404

			Returns: The updated tag object. Status: 200 OK, 400 Bad Request (missing name), 404 Not Found.	Not Found.
/api/ingredients	Description: Searches for ingredients. Supports a query parameter q for partial name matching. Expects: Optional query string ?q=search_term. Returns: Array of matching ingredient objects. Status: 200 OK.	Description: (Admin/Mod only) Adds a new ingredient to the global list. Expects: { ingredient_name: string }. Returns: The created ingredient object. Status: 201 Created, 400 Bad Request.	**-**	**-**
/api/ingredients/:id	Description: Fetches a single ingredient's details. Returns: Ingredient object.	-	Description: (Admin/Mod only) Updates the name of an ingredient. Expects: {	Description: (Admin/Mod only) Deletes an ingredient. Prevents deletion if the ingredient is currently linked to any recipes.

	Status: 200 OK, 404 Not Found.		ingredient_name: string }. Returns: The updated ingredient object. Status: 200 OK, 404 Not Found.	Returns: Empty body. Status: 204 No Content, 404 Not Found, 409 Conflict (in use).
/api/stats/categories/popular	Description: (Admin/Mod only) Returns a list of categories ranked by the number of recipes assigned to them. Returns: Array of objects: { categoryId, categoryName, recipeCount }. Status: 200 OK.	-	-	-
/api/stats/categories/ratings	Description: (Admin/Mod only) Returns categories ranked by their average recipe ratings. Returns: Array of objects: {	-	-	-

	categoryId, categoryName, avgRating, ratingsCount }. Status: 200 OK.			
/api/recipes	Description: A powerful search endpoint. Supports pagination and filtering by text, category, tags, ingredients, and minimum rating. Can sort by newest, rating, or popularity. Expects: Query params: page, pageSize, textSearch, categoryId, tags (comma-separated), ingredientSearch, minRating, sortBy, sortDir. Returns: Paginated object { items: Recipe[], total, page, pageSize }. Status: 200 OK.	Description: (Authenticated) Creates a new recipe. Automatically handles transaction logic for saving ingredients and tags. Normalizes step images and cropping data. Expects: { title, description, categoryId, steps: [], tags: [], ingredients: [], imageUrl, imageCrop }. Returns: The fully populated new recipe object.	-	-

/api/recipes/mine	Description: (Authenticated) Retrieves recipes created by the logged-in user with text search support. Returns: Paginated object of user-owned recipes. Status: 200 OK, 401 Unauthorized.			
/api/recipes/:id	Description: Fetches full details for a single recipe, including full ingredient list, tags, favorite count, and rating count. Returns: Detailed recipe object. Status: 200 OK, 404 Not Found	-	Description: (Owner/Admin/Mod only) Updates an existing recipe. If an Admin/Mod edits someone else's recipe, an audit log is generated. Expects: JSON body with updated recipe fields. Returns: The updated detailed recipe object. Status: 200 OK, 403 Forbidden, 404 Not Found.	Description: (Owner/Admin/Mod only) Permanently removes a recipe and logs the action. Returns: Empty body. Status: 204 No Content, 403 Forbidden, 404 Not Found.

/api/recipes/users/:userid	Description: Retrieves all recipes authored by a specific user. Returns: Array of recipe objects. Status: 200 OK.	-	-	-
/api/recipes/tags/:tagid	Description: Retrieves all recipes associated with a specific tag ID. Returns: Array of recipe objects. Status: 200 OK.	-	-	-
/api/recipes/categories/:categoryid	Description: Retrieves all recipes belonging to a specific category ID. Returns: Array of recipe objects. Status: 200 OK.	-	-	-
/api/recipes/:id/pdf	Description: Generates and streams a PDF	-	-	-

	version of the recipe for download/printing. Returns: A binary PDF stream. Status: 200 OK, 404 Not Found.			
/api/users	Description: (Admin/Mod only) Retrieves a list of all registered users, excluding sensitive fields like password hashes. Returns: Array of user objects { user_id, username, email, role_id, created_at, is_active }. Status: 200 OK.	Description: Registers a new user account. Passwords are automatically hashed before storage. Defaults new users to role_id: 3 (likely "Standard User"). Expects: { username, email, password, role_id? }. Returns: The new user object (no password).	**-**	**-**
/api/users/:id	Description: Fetches profile information for a specific user. Returns: User object { user_id, username, email, role_id,	-	Description: (Admin only) Updates user details, including role and active status. Generates a descriptive audit log detailing exactly what was changed.	Description: (Admin only) Permanently removes a user account from the system. Returns: Empty body.

	is_active }. Status: 200 OK, 404 Not Found.		Expects: { username, email, role_id, is_active }. Returns: The updated user object. Status: 200 OK, 400 Bad Request, 404 Not Found.	Status: 204 No Content, 404 Not Found.
/api/auth or /api/auth/login	-	Description: Authenticates a user using a username (or email) and password. On success, it establishes a persistent session via a cookie and logs a success audit event. Expects: { "username": "...", "password": "..." }. Returns: User profile and assigned role IDs. Status: 200 OK, 401 Unauthorized (invalid credentials), 400 Bad Request.	**-**	**-**
/api/auth or /api/auth/logout	-	-	**-**	Description: Terminates the current user session and

				clears the session cookie. Returns: { "message": "Logged out" }. Status: 200 OK.
/api/auth or /api/auth/me	Description: "Who Am I" endpoint. Checks the session cookie to see if a user is currently logged in. Returns: User profile and roles if logged in; { user: null, roles: null } if not. Status: 200 OK.	-	-	-
/api/auth/register	-	Description: Creates a new user account, hashes the password, and automatically logs them in immediately after successful creation. Expects: { username, email, password, role_id? }. Returns: The new user profile and a success	-	-

		message.		
/api/upload	-	Description: Uploads a single file to a specified directory. If no file is provided in the request but a name and path are, the system attempts to delete the existing file at that location. Auth: Requires any valid user role (1, 2, 3, or 4). Expects: multipart/form-data with fields: file (the binary), path (subfolder), and name (desired filename). Status: 200 OK, 413 Payload Too Large, 400 Bad Request.	**-**	**-**