Group 25 Code Summarizer API Documentation

Endpoints

Data types are in **bold** and further descriptions can be found in the **data types** table.

Name	**Parameters**	**Usage**
GET /login	Headers: username, password	Endpoint used to login from frontend. Username & password are compared to values stored in the database. If valid values are provided, an **access token** is returned.
GET /createAccount	Headers: newusername, newpassword, newadmin, **token** (optional)	This endpoint will make a new account with the provided credentials. If newadmin is “true”, then a valid **token** for an administrator must be provided. Otherwise, no token is needed. This ensures that anyone can make an account, but only admins can create admin accounts.
POST /summarize	Headers: username, **token** Body: JSON object with properties: code, models. Model is a string, containing a comma separated list of **model names**, without spaces.	This endpoint will take in provided code, and produce LLM generated summaries of it by each model listed in the “models” property of the body. The return value is an array, where each entry is a **completion,** except for the final object which is of the form {id: id}. This id can be used in calls to the /setRating endpoint.

GET /getSummarizations	Headers: username, **token,** targetUsername	This endpoint provides all of a user's previous summaries and ratings. The return value is an array of objects, where each object represents one of the user's code-summary pairs. Each object has the property “id” (which can be used as a parameter to the /setRating endpoint), and “code”, which is the code the user requested a summary for. The final property is completions**,** which is an array of **completion** objects, containing the LLM summaries. If targetUsername isn’t the same as username, the request will be treated as if a user is attempting to get another user’s past summaries. In order to do this, an admin token must be provided. For the sake of simplicity, the **completion** and corresponding **rating** objects are combined into the same entry in the returned array, so that iterating through multiple arrays in the frontend isn’t necessary. This combined object simply has all of the properties of both the **completion** and **rating** types. To clarify, the response is an array of objects of the form: {id: Number, code: String, completions: [combined **completion** and **rating** object]} The goal of this endpoint is to allow the frontend to display a user's past summarizations,

		and provide them with the means to rate them.
POST /uploadSummarization	Headers: username, **token** Body: JSON object with properties: code, completions, ratings. The “code” property is a string containing the summarized code. The completions property is an array of **completions.** The ratings property is an array of **ratings.**	This endpoint allows for the frontend to give users the capability to upload code-summary pairs, with ratings. Some notes: Because each rating corresponds to an individual completion, an equal amount of ratings & completions should be provided. Additionally, ratings should be supplied in the same order that the completions were provided in. For example, the rating for the first completion in the completions array should also be the first element in the ratings array.
POST /setRating	Headers: username, **token,** id Body: JSON object with one property, “ratings” – an array of **rating** objects.	This endpoint allows for setting the ratings of a particular code-summarizations combo, specified by the provided ID. Like the uploadSummarization endpoint, ratings will be applied to corresponding completions based on their index in their respective arrays. It’s worth noting that while “completions” isn’t a parameter of this endpoint, a frontend implementation would still need access to the completions array for the summarization being rated, so that the ratings can be supplied in the proper order. It is expected that prior to using the /setRating endpoint, the /getSummarizations or /summarize endpoints will have been called first, thus providing access to the

		completions array.
GET /stats	Headers: username, **token** In this case, the username and token should correspond to an admin account.	Allows an admin user to view stats about the service. Returns a single **stats** object.
GET /checkAdmin	Headers: username, **token**.	Allows for the frontend to check whether a valid token corresponds to an admin account or not. Responds with a JSON object with one boolean property, “isAdmin”.
GET /getUsers	Headers: username, **token.** Username and token should correspond to an admin account.	Allows for admins to retrieve a list of all of the users stored in the database. Returns a JSON object with the “userList” property, which is an array of every username.
GET /makeAdmin	Headers: username, token, adminName Username and token should correspond to an admin account.	This endpoint allows an admin user to make another user (specified by “adminName” header) into an admin. The provided adminName must be the name of an existing user.
GET /deleteUser	Headers: username, **token**, toDelete Username and token should correspond to an admin account.	Allows for an admin to delete a user, with the username specified by “toDelete”. Users must exist in the database in order to be deleted.
GET /changePassword	Headers: username, token, newPassword, targetUsername	Allows for a user to change their password to a new value, “newPassword”. It requires that the user is already logged in and therefore has a valid token. Otherwise, they will need to ask an admin to reset their password for them. If the user isn’t an admin, the “username” and

		“targetUsername” fields must be equal, since a non-admin cannot change the password of other accounts. Additionally, the provided token must correspond to the user’s username. Additionally, it can be used by admins to change the password of another account, specified by the “targetUsername” parameter. If **any** of these required conditions aren’t met, the response code will be 401. On success, a 400 status code will be sent.

Data Types

Name	**Format**	**Use**
Token	String	The token is a string required for all endpoints besides /login and /createAccount. It is used to prove that the user has the required permissions to use an endpoint. It is a string generated from the base64 encoding of a SHA256 hash of the user’s username padded with random values. It is provided to the frontend upon a successful login.
Model names	String containing a concatenation of comma-separated valid model names. The valid models are: gpt-4-turbo, gpt-3.5-turbo, claude-3-opus-20240229, claude-3-haiku-20240307. An example of a valid model names-list would be: “gpt-4-turbo,gpt-3.5-turbo”.	Used as a parameter to the /summarize endpoint to specify which models to use when generating summaries.

Completion	Object with two properties, “model” and “text”. Model specifies which model (from **model names**) generated this summary. Text contains the LLM generated summary.	This object is returned by numerous endpoints and provides the frontend with the summaries to display to the user.
Rating	An object with properties: naturalRating, usefulRating, and consistentRating, favorite, userNotes. Each rating property can take a value 1-5 and represents a user’s opinion of a summary. Favorite is a bool indicating whether this was the user’s favorite summary. Finally, userNotes is a string containing the comments the user had about this summary.	This object is returned by the getSummaries endpoint as part of the completion object to show the existing ratings for a given summary in the database. This object is attached as a parameter to the uploadSummarization and setRating endpoints to assign ratings to summaries.