GREEN does not require authorization token in header

Service	Endpoints	Description	Request	Response
Predict	POST /predict	Get bounding boxes	Body: multipart/form-data “width”: <number> “image”: <file: image/jpeg>	{ "boxes": [{ "x": <number>, "y": <number>, "w": <number>, "h": <number>, "confidence": <number>, "class": "hold" | “volume” }, ...], “width”: <number>, “height: :<number }
User	POST /user/signup	Sign ups and send verification email	Body: { "name": <str>, "email": <str>, "password": <str> }	{ “Message”: “Sign up success”, "Destination": "y***@g***.com", "DeliveryMedium": "EMAIL", "AttributeName": "email" }
	POST /user/confirm	Confirms sign up with verification	Body: { "name": <str>, "code” <str> }	{ "Message” : “Confirmation success” }
	POST /user/login	Returns access token	Body: { “email”: <str>, “password”: <str> }	{ “Message”: “Sign in success”, "AccessToken": <str>, "ExpiresIn": <int:seconds>, "TokenType": "Bearer", “RefreshToken”: <str>, “IdToken”: <str>, }
	POST /user/logout	Invalidates access token	**Bearer Authorization. IE: Ensure header contains** Bearer <str:accessToken>	{ "Message” : “Sign out success” }
Route	POST /route/new	Create new route by saving to S3 bucket Only allow if the gym is valid	**Bearer Authorization. IE: Ensure header contains** Bearer <str:accessToken> Body: multipart/form-data “country”: <str> “expiredTime”: <str: ISOFormat> “routeName”: <str> “gymLocation”: <str: LatLong> “ownerGrade”: <number> “routePhoto”: <file: JPG, 5MB>	{ "Message": "Create route success", "Item": { "username": <str:routeowner>, "createdAt": <str>, "expiredTime": <str>, “routeName”: <str>, "gymLocation": <str>, "routeURL": <str: url>, "ownerGrade": <number>, "publicGrade": <number>, "publicGradeSubmissions": [{ "userEmail": <number> }, ...], "vote": <number>, "upVotes": [], "reports": [], "commentCount": 0, "comments": [] } }

	POST /route/details	Get photo, votes, grades, comments of one route (Returns slightly different data depending on if authorization header is provided)	Body: { “username": <str:routeowner>, "createdAt": <str> } # Acts as route identifier **<u>Optional</u>** **Bearer Authorization. IE: Ensure header contains** Bearer <str:accessToken> Including this will check if the user has voted, graded, reported the route. The default values are false.	{ "Message": "Get route details success", "Item": { "username": <str:routeowner>, "createdAt": <str>, "expiredTime": <str>, “routeName” :<str>, "gymLocation": <str>, "routeURL": <str>, "ownerGrade": <number>, "publicGrade": <number>, "vote": <number>, "comments": [], "hasVoted": <boolean>, "hasReported": <boolean>, "hasGraded": <boolean>, "graded": <number> } }
	POST /route/details/vote	Upvote route	**Bearer Authorization. IE: Ensure header contains** Bearer <str:accessToken> Body: { “username": <str:routeowner>, "createdAt": <str> }	{ "Message": "Upvote route success" }
	POST /route/details/grade	Assign grade to route (Handles differently for owner and public in backend)	**Bearer Authorization. IE: Ensure header contains** Bearer <str:accessToken> Body: { “username": <str:routeowner>, "createdAt": <str>, “grade”: <number> }	{ "Message": "Grade route success", "Item": { "publicGrade": <number>, "ownerGrade": <number> } }
	POST /route/details/comment	Add a new comment	**Bearer Authorization. IE: Ensure header contains** Bearer <str:accessToken> Body: { “username": <str:routeowner>, "createdAt": <str>, “comment”: <str: not more than 150 chars> }	{ "Message": "Comment route success", "Item": { "username": <str>, "timestamp": <number: unix>, "comment": <str> } }
	DELETE /route/details/comment	Delete the comment if you are poster or route owner	**Bearer Authorization. IE: Ensure header contains** Bearer <str:accessToken> Body: { “username": <str:routeowner>, "createdAt": <str>, "timestamp": <number: unix> }	{ "Message": "Delete comment success" }
	POST /route/details/report	Report route	**Bearer Authorization. IE: Ensure header contains** Bearer <str:accessToken>	{ "Message": "Report route success" }

			Body: { “username": <str:routeowner>, "createdAt": <str> }	
	GET /route/all	Get routes by gyms Note: Frontend call another API to determine proximity	Url parameter of “gymLocation”	{ "Message": "Query routes by gym success", "Items": [{ “routeName”: <str>, "publicGrade": <number>, "routeURL": <str:url>, "commentCount": 0, "username": <str>, "gymLocation": <str>, "createdAt": <str>, "vote": <number> }] }
	GET /route/gym/all	Get gyms	If no parameter, returns all the gyms **Or** If url parameter of “country” exists, returns the gyms in the country (If there are 100+ gyms this would be preferred)	{ "Message": "Scan all gyms success", "Items": [{ "country": "Singapore", "gymLocation": <str>", "gymName": <str> }] }
	DELETE /route	Delete a particular route if you are route owner	**Bearer Authorization. IE: Ensure header contains** Bearer <str:accessToken> Body: { "createdAt": <str: ISOString> }	{ "Message": "Delete route success" }

Database supported features

- Get all gyms and their locations
- Search routes by gym
- Add route by gym
- Upvote
- Report
- Upgrade / Downgrade
- View Owner Grade & Public Grade
- View votes
- View route photo
- Search routes by a single username
- Batch removal of outdated routes
- ADMIN: Add gym
- ADMIN: Delete route
- ADMIN: Delete gym

(Trying to do Single Table style which is recommended by AWS)

Database	Fields	Index

Route	username **(PK)** <cognito jwt token username> createdAt **(SK)** yyyy-mm-dd**T**hh:mm:ss**Z** expiredTime <time to live datetime> routeName <str> gymLocation locational_string routeURL s3bucketlink -> /<emailHash>/<createdAt>/ ownerGrade <integer> publicGrade (Includes ownerGrade) <integer> publicGradeSubmissions [{ email: <str>, grade:<integer> }, { }, …] vote <integer> upVotes [list of emails] reports [list of emails] commentCount <integer> comments [{ username: <integer>, comment: <str>, timestamp: <number> }, { }, …]	Global Secondary Index gymLocation **(PK)** createdAt **(SK)** routeName (projected) publicGrade (projected) username (projected) vote (projected) commentCount (projected) routeURL (projected)
Gym	country (PK) <str> (Ensure only alphanumeric and space) gymLocation (SK) <str:latLong> (Google Maps right click) gymName <str>	