

**GREEN does not require authorization token in header**

Service	Endpoints	Description	Request	Response
Predict	POST /predict	Get bounding boxes	Body: <u>multipart/form-data</u> "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: <u>application/json</u> { "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: <u>application/json</u> { "name": <str>, "code" <str> }	{ "Message": "Confirmation success" }
	POST /user/login	Returns access token	Body: <u>application/json</u> { "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	<b>Bearer Authorization. IE: Ensure header contains</b>  Bearer <str:accessToken>	{ "Message": "Sign out success" }
	DELETE /user/delete	Deletes Cognito User	<b>Bearer Authorization. IE: Ensure header contains</b>  Bearer <str:accessToken>	HTTP 204
Route	POST /route/new	Create new route by saving to S3 bucket  Only allow if the gym is valid	<b>Bearer Authorization. IE: Ensure header contains</b>  Bearer <str:accessToken>  Body: <u>multipart/form-data</u>  "countryCode": <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> }, ... ], "voteCount": <number>, "upVotes": [], } }

			<pre>"reports": [], "commentCount": 0, "comments": [] } }</pre>
POST /route/details	Get photo, votes, grades, comments of one route  (Returns slightly different data depending on if authorization header is provided)	Body: <u>application/json</u> { "username": <str:routeowner>, "createdAt": <str> } # Acts as route identifier  <u>Optional</u>  <b>Bearer Authorization. IE: Ensure header contains</b>  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>, "displayName": <str>, "expiredTime": <str>, "routeName": <str>, "gymLocation": <str>, "routeURL": <str>, "ownerGrade": <number>, "publicGrade": <number>, "voteCount": <number>, "comments": [], "hasVoted": <boolean>, "hasReported": <boolean>, "hasGraded": <boolean>, "graded": <number> } }
POST /route/details/vote	Upvote route	<b>Bearer Authorization. IE: Ensure header contains</b>  Bearer <str:accessToken>  Body: <u>application/json</u> { "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)	<b>Bearer Authorization. IE: Ensure header contains</b>  Bearer <str:accessToken>  Body: <u>application/json</u> { "username": <str:routeowner>, "createdAt": <str>, "grade": <number> }	{ "Message": "Grade route success", "Item": { "publicGrade": <number>, "ownerGrade": <number> } }
POST /route/details/comment	Add a new comment	<b>Bearer Authorization. IE: Ensure header contains</b>  Bearer <str:accessToken>  Body: <u>application/json</u> { "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	<b>Bearer Authorization. IE: Ensure header contains</b>  Bearer <str:accessToken>  Body: <u>application/json</u> { "username": <str:routeowner>, "createdAt": <str>, }	{ "Message": "Delete comment success" }

			"timestamp": <number: unix> }	
POST /route/details/report	Report route		<b>Bearer Authorization. IE: Ensure header contains</b>  Bearer <str:accessToken>  Body: <u>application/json</u> { "username": <str:routeowner>, "createdAt": <str> }	{ "Message": "Report route success" }
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>, "voteCount": <number> } ] }
GET /route/gym/all	Get gyms		If no parameter, returns all the gyms  <u>Or</u>  If url parameter of "countryCode" exists, returns the gyms in the countryCode (If there are 100+ gyms this would be preferred)	{ "Message": "Scan all gyms success", "Items": [ { "countryCode": "Singapore", "gymLocation": <str>, "gymName": <str> } ] }
DELETE /route	Delete a particular route if you are route owner		<b>Bearer Authorization. IE: Ensure header contains</b>  Bearer <str:accessToken>  Body: <u>application/json</u> { "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 vote count
- View route photo
- Search routes by a single username
- TTL removal of outdated routes
- ADMIN: Add gym
- ADMIN: Delete route
- ADMIN: Delete gym

Database	Fields	Index
Route	username ( <b>PK</b> ) <cognito jwt token username> createdAt ( <b>SK</b> ) yyyy-mm-ddThh:mm:ssZ ttl <time to live datetime> routeName <str> gymLocation locational_string routeURL s3bucketlink -> /<emailHash>/<createdAt>/ ownerGrade <integer> publicGrade (Includes ownerGrade) <integer> publicGradeSubmissions [{ email: <str>, grade:<integer> }, { }, ...] voteCount <integer> upVotes [list of emails] reports [list of emails] commentCount <integer> comments [{ username: <integer>, comment: <str>, timestamp: <number> }, { }, ...]	Global Secondary Index gymLocation ( <b>PK</b> ) createdAt ( <b>SK</b> ) routeName (projected) publicGrade (projected) username (projected) voteCount (projected) commentCount (projected) routeURL (projected)
Gym	countryCode (PK) <str> (Ensure only alphanumeric and space) gymLocation (SK) <str:latLong> (Google Maps right click) gymName <str>	