

API Specification Document

API Specification Document.....	1
1. Authentication & User Management	4
1.1 Create User Account.....	4
1.2 Update User Profile.....	5
1.3 Get User Profile	6
1.3 Login.....	8
1.4 Reset Password Request.....	9
1.5 Reset Password (ValidateToken)	10
1.6 (Optional) Refresh Token	11
2. Theory of Change Management (NoSQL).....	12
2.1 Create Project Name.....	12
2.2 Rename Project	14
2.3 Save TOC (Update TOC content in existing project).....	16
2.2 Get TOC	19
2.3 (Optional) Delete TOC	22
2.4 (Optional) List user TOCs.....	23
3. Payment & Subscription Management.....	25
3.1 Create Payment.....	25
3.2 Confirm Payment.....	25
3.3 Get Subscription Status	25
3.4 Cancel Subscription.....	26
3.5 (Need?) Update Subscription.....	26
Invoice Data Object Needs with APIs'.....	26
4. Admin Dashboard.....	27
4.1 Get Subscription User Details	27
4.2 Update Subscription Details	27
4.3 Get Registered User Details	27
4.4 (Need?) Get Revenue	27
Need Plan Create/Update/Delete APIs.....	27
5. (Optional) Export & File Management	28

5.1 Export TOC as PDF	28
5.2 Export TOC as PNG	28
5.3 Get Download URL.....	28
6. (Optional) Help Guide & Content Management.....	29
6.1 Get Help Content.....	29
6.2 Get Help Section.....	29
7. (Optional) Session Management	30
7.1 Save Session Data	30
7.2 Get Session Data	30
8. (Optional) Email Service	30
8.1 Send Email (Internal Use)	30
9. (Optional) Health & Monitoring.....	30
9.1 Health Check	30
Parameter Details	30

Service Component	List of API
Authentication & User Management	Create User Account Update User Profile Get User Profile Login Reset Password Request Reset Password (ValidateToken) (Optional) Refresh Token
Theory of Change Management (NoSQL)	Create Project Rename Project Save/Update TOC Get TOC/ List user TOCs Delete TOC
Payment & Subscription Management	Create Payment Confirm Payment Get Subscription Status Cancel Subscription (Need?) Update Subscription
Admin Dashboard	Get Subscription User Details Update Subscription Details Get Registered User Details (Need?)Get Revenue (Need?) Traffic Analytics
(Optional) Export & File Management	Export TOC as PDF Export TOC as PNG Get Download URL
(Optional) Help Guide & Content Management	Get Help Content Get Help Section
(Optional) Session Management	Save Session Data Get Session Data
(Optional) Email Service	Send Email (Internal Use)
(Optional) Health & Monitoring	Health Check

BaseURL : <http://toc-user-backend.vercel.app>

QFO --> GitHub --> <https://github.com/QualityForOutcomes>

QFO --> Vercel --> <https://vercel.com/quality-of-outcomes-projects>

1. Authentication & User Management

1.1 Create User Account

API URL	POST {BASE_URL}/api/user/Create
Request	<pre>{ "email": "string", "password": "string", "firstName": "string", "lastName": "string", "organization": "string", "username": "string", "acceptTandC": "boolean", "newsLetterSubs": "boolean" }</pre>
Response	<pre>{ "success": boolean, "data": { UserData}, "statusCode": "HTTPStatus" }</pre>
Example	<pre>curl --location 'http://nodejs-serverless-function-express-rho-ashen.vercel.app/api/user/Create' \ --header 'Content-Type: application/json' \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJmVtYVYjoiENp6ekBkLmNvbSIsInVzZXJJZCI6MTQsImh dCI6MTc1ODYyNTU5OSwiZXhwIjoxNzU4NzExOTk5fQ.elaS_DVEgAiSPOpvc5GgcXaNaYjD37u-jSBq9DzgrEs' \ --data-raw '{ "email": "zzzz@d.com", "password": "SecurePass123!", "username": "zzz", "firstName": "Aa", "lastName": "Bb", "organisation": "ABC" }' { "success": true, "message": "User registered successfully", "data": { "userId": 15, "email": "zzzz@d.com", "username": "zzz", "firstName": "Aa", "lastName": "Bb", "organisation": "ABC", "avatarUrl": null, "displayName": "Aa Bb", "createdAt": "2025-09-23T11:10:57.004" }, "statusCode": 201 } { "success": false, "message": "Validation failed", "error": ["Valid email is required"], "statusCode": 400 }</pre>

The image displays two screenshots of a REST client interface, likely Postman, showing the results of a POST request to a user creation endpoint.

Top Screenshot (Successful Request):

- URL:** `http://([BASE_URL])User/Create`
- Method:** POST
- Body (JSON):**

```
{
  "email": "zzzz@com",
  "password": "SecurePass123!",
  "username": "zzz",
  "firstName": "Aa",
  "lastName": "Bb",
  "organisation": "ABC"
}
```
- Status:** 201 Created
- Response (JSON):**

```
{
  "success": true,
  "message": "User registered successfully",
  "data": {
    "userId": 15,
    "email": "zzzz@com",
    "username": "zzz",
    "firstName": "Aa",
    "lastName": "Bb",
    "organisation": "ABC",
    "accessToken": null,
    "displayName": "Aa Bb",
    "createdAt": "2025-09-20T11:10:57.004"
  },
  "statusCode": 201
}
```
- Terminal Output:**

```
1 curl --location 'http://
nodejs-serverless-function-express-rho-
ashen-vece1.api/user/Create' \
2 --header 'Content-Type: application/json' \
3 --header 'Authorization: *****' \
4 --data '{
5   "email": "zzzz@com",
6   "password": "SecurePass123!",
7   "username": "zzz",
8   "firstName": "Aa",
9   "lastName": "Bb",
10  "organisation": "ABC"
11 }'
```

Bottom Screenshot (Failed Request):

- URL:** `http://([BASE_URL])User/Create`
- Method:** POST
- Body (JSON):**

```
{
  "email": "zzzzd.com",
  "password": "SecurePass123!",
  "username": "zzz",
  "firstName": "Aa",
  "lastName": "Bb",
  "organisation": "ABC"
}
```
- Status:** 400 Bad Request
- Response (JSON):**

```
{
  "success": false,
  "message": "Validation failed",
  "errors": [
    "Valid email is required"
  ],
  "statusCode": 400
}
```
- Terminal Output:**

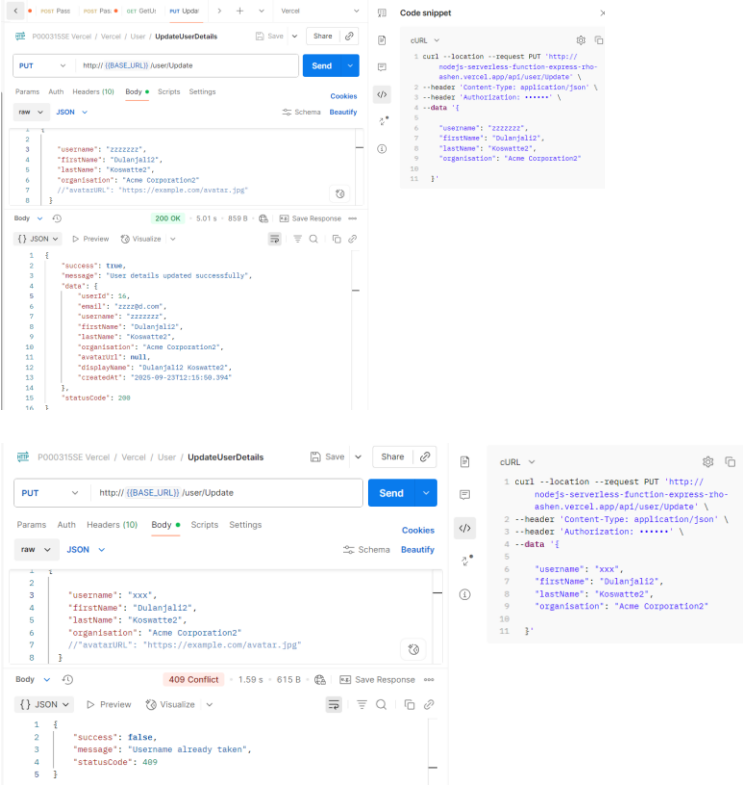
```
1 curl --location 'http://
nodejs-serverless-function-express-rho-
ashen-vece1.api/user/Create' \
2 --header 'Content-Type: application/json' \
3 --header 'Authorization: *****' \
4 --data '{
5   "email": "zzzzd.com",
6   "password": "SecurePass123!",
7   "username": "zzz",
8   "firstName": "Aa",
9   "lastName": "Bb",
10  "organisation": "ABC"
11 }'
```

1.2 Update User Profile

API URL	PUT {BASE_URL}/api/user/Update
Request	{ "firstName": "string", "lastName": "string", "organization": "string", “username”:”string” }
Response	{ "success": true, "data": { UserData }, “statusCode”:”HttpStatus” }
Example	<pre>curl --location --request PUT 'http://nodejs-serverless-function-express-rho-ashen.vercel.app/api/user/Update' \ --header 'Content-Type: application/json' \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWwiOiIxNCIsImVtYWlsIjoienp6ekBkLmNvbSIsInVzZXJJZCMTQsImhh dCI6MTc1ODYyNTU0SWiZXhwIjoxNzU4NzExOTk5fQ.elaS_DVEgAiSPOPvc5GgcXaNAYjD37u-jSBq9DzgrEs' \ --data '{ "username": "zzzzzz", "firstName": "Dulanjali2", "lastName": "Koswatte2", "organisation": "Acme Corporation2" }'</pre>

```
{
  "success": true,
  "message": "User details updated successfully",
  "data": {
    "userId": 16,
    "email": "zzzz@d.com",
    "username": "zzzzzz",
    "firstName": "Dulanjali2",
    "lastName": "Koswatte2",
    "organisation": "Acme Corporation2",
    "avatarUrl": null,
    "displayName": "Dulanjali2 Koswatte2",
    "createdAt": "2025-09-23T12:15:50.394"
  },
  "statusCode": 200
}

{
  "success": false,
  "message": "Username already taken",
  "statusCode": 409
}
```



1.3 Get User Profile

API URL	GET {{BASE_URL}}/api/user/Get
Response	{ "success": true, "data": { "userId": "string",

```

    "firstName": "string",
    "lastName": "string",
    "organization": "string",
    "email": "string",
    "username": "string"
  }
}

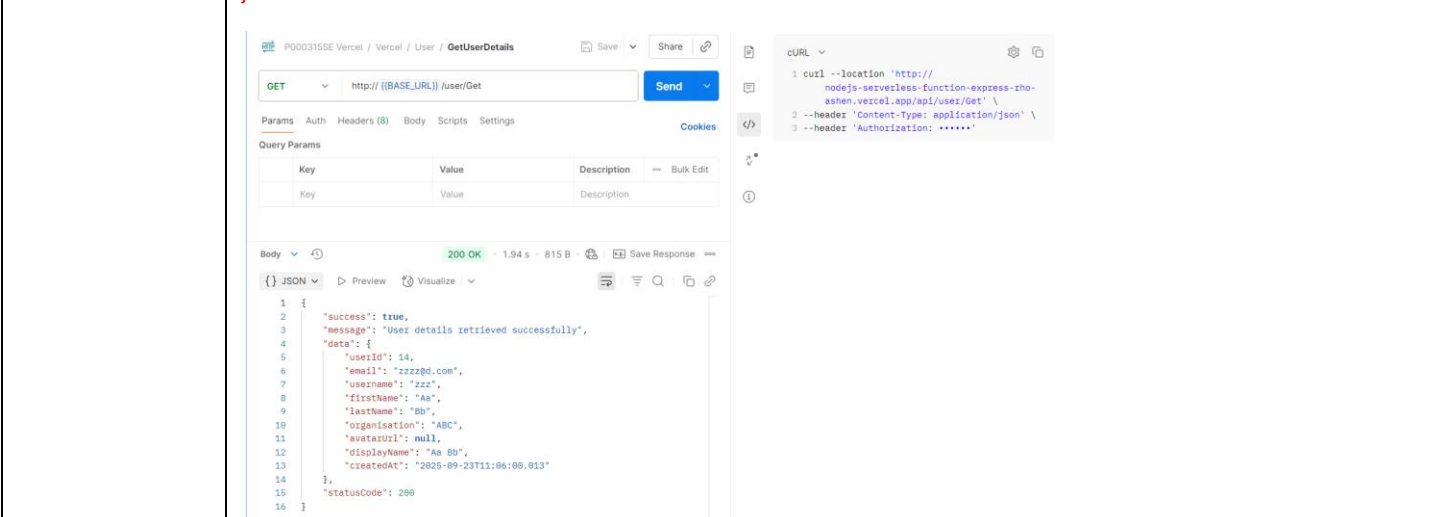
```

Example	<code>curl --location 'http://nodejs-serverless-function-express-rho-ashen.vercel.app/api/user/Get' \</code> <code>-H 'accept: application/json'</code>
---------	--

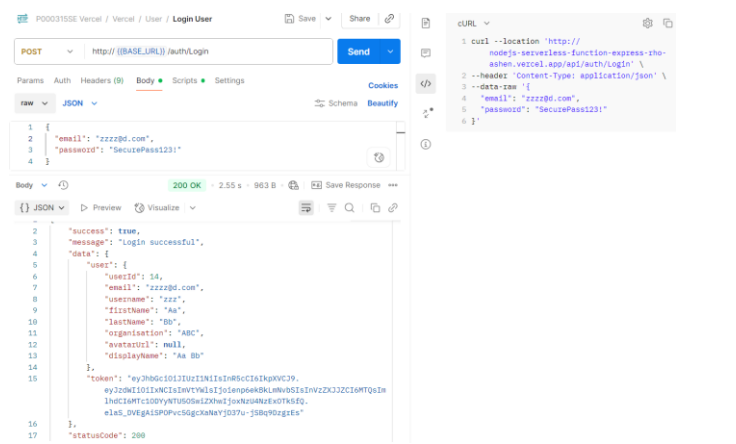
```
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM5MTZ0eXNpdjEyZW5ka2UiOiJ1bm90b290LmNvbSI6ImNvZDZzgrEs'
dCI6MTc1ODYyNTU0S0wiZXhwIjoxNzU4NzExOTk1fQ.eyJ0eXNpdjEyZW5ka2UiOiJ1bm90b290LmNvbSI6ImNvZDZzgrEs'
```

```
{
  "success": true,
  "message": "User details retrieved successfully",
  "data": {
    "userId": 14,
    "email": "zzzz@d.com",
    "username": "zzz",
    "firstName": "Aa",
    "lastName": "Bb",
    "organisation": "ABC",
    "avatarUrl": null,
    "displayName": "Aa Bb",
    "createdAt": "2025-09-23T11:06:00.013"
  },
  "statusCode": 200
}

{
  "success": false,
  "message": "User not found",
  "statusCode": 404
}
```



1.3 Login

API URL	POST {{BASE_URL}}/api/auth/login
Request	<pre>{ "email": "string", "password": "string" }</pre>
Response	<pre>{ "success": true, "data": { UserData }, "statusCode": "HttpSttaus" }</pre>
Example	<pre>curl --location 'http://nodejs-serverless-function-express-rho-ashen.vercel.app/api/auth/Login' \ --header 'Content-Type: application/json' \ --data-raw '{ "email": "zzzz@d.com", "password": "SecurePass123!" }'</pre> <pre>{ "success": true, "message": "Login successful", "data": { "user": { "userId": 14, "email": "zzzz@d.com", "username": "zzz", "firstName": "Aa", "lastName": "Bb", "organisation": "ABC", "avatarUrl": null, "displayName": "Aa Bb" }, "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxNCIsImVtYWlsIjoienp6ekBkLmNvbSIsInVzZXJJZCI6MTQsImIhZCI6MTc1ODYyNTU5OSwiZXhwIjozNzU0NzExOTk5fQ.elaS_DVEgAiSPOPvc5GgcXaNaYjD37u-jSBq9DzgrEs" }, "statusCode": 200 }</pre> <pre>{ "success": false, "message": "Invalid email or password", "statusCode": 401 }</pre> 

The screenshot displays the Burp Suite interface for a REST client. The top bar shows the project name 'POOD3155E Vector' and the user 'User / Login User'. The main panel is divided into several sections:

- Request Section:**
 - Method:** POST
 - URL:** http://(BASE_URL)/auth/login
 - Params:** Auth, Headers (0), Body, Scripts, Settings
 - Body:** JSON (selected)
- Request Body:**

```
1 {
2   "email": "zzzz286.com",
3   "password": "SecurePass123!"
4 }
```
- Response Section:**
 - Status:** 401 Unauthorized
 - Size:** 187 B
 - Save Response:** (button)
- Response Body:**

```
1 {
2   "success": false,
3   "message": "Invalid email or password",
4   "statusCode": 401
5 }
```
- Right Panel:**
 - cURL:** curl -X POST -H 'Content-Type: application/json' -d '{"email": "zzzz286.com", "password": "SecurePass123!"}' http://(BASE_URL)/auth/login

1.4 Reset Password Request

API URL	POST {{BASE_URL}}/api/auth/password.Reset
Request	{ "email": "string", "action": "request-reset" }
Response	{ "success": true, "message": "Reset password email sent", "statusCode": "HttpStatus" }
Example	<pre>curl --location 'http://nodejs-serverless-function-express-rho-ashen.vercel.app/api/auth/password.Reset' \ --header 'Content-Type: application/json' \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWUiOiIxMjYyNTU0ODYyNTU0SOSwiZSwiaWF0IjoxNzU4NmExOTk5fQ.elaS_DVEgAiSPOPvc5GgcXaNyD37u-jSBq9DzgrEs' \ --data-raw '{ "email": "s4084228@student.rmit.edu.au", "action": "request-reset" }' { "success": true, "message": "If this email exists, you will receive a reset code", "data": {}, "statusCode": 200 } { "success": false, "message": "Validation failed", "error": ["Invalid action. Must be \"request-reset\" or \"verify-token\""], "statusCode": 400 }</pre>

POST

http://[BASE_URL]/auth/password.Reset

Send

Params

Auth

Headers (9)

Body

Scripts

Settings

Cookies

raw

JSON

Schema

Beautify

```
1 {
2   "email": "s4084220@student.zmit.edu.au",
3   "action": "request-reset"
4 }
```

Body

200 OK

3.59 s

648 B

Save Response

JSON

Preview

Visualize

```
1 {
2   "success": true,
3   "message": "If this email exists, you will receive a reset code",
4   "data": {},
5   "statusCode": 200
6 }
```

1 curl --location 'http://
nodejs-serverless-function-express-rho-
ashen.vercel.app/api/auth/password.
Reset' \
2 --header 'Content-Type: application/json' \
3 --header 'Authorization: *****' \
4 --data-raw '{
5 "email": "s4084220@student.zmit.edu.au",
6 "action": "request-reset"
7 }'

P0003155E Vercel / ... / User / Password-Reset SendEmail

Save

Share

POST

http://[BASE_URL]/auth/password.Reset

Send

Params

Auth

Headers (9)

Body

Scripts

Settings

Cookies

raw

JSON

Schema

Beautify

```
1 {
2   "email": "s4084220@student.zmit.edu.au",
3   "action": "request-reset"
4 }
```

Body

400 Bad Request

406 ms

650 B

Save Response

JSON

Preview

Visualize

```
1 {
2   "success": false,
3   "message": "Validation failed",
4   "errors": [
5     "Valid email is required"
6   ],
7   "statusCode": 400
8 }
```

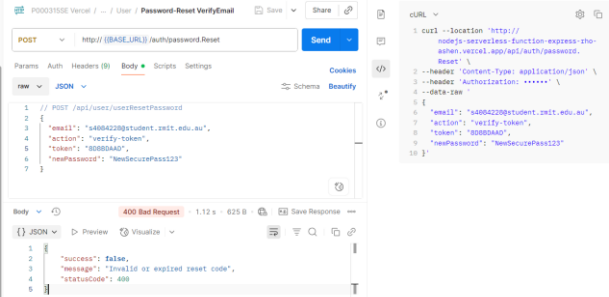
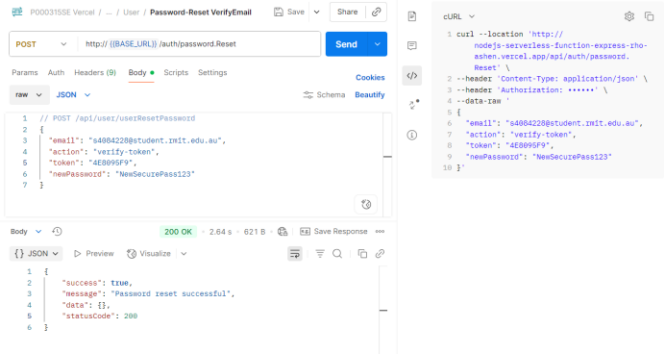
1 curl --location 'http://
nodejs-serverless-function-express-rho-
ashen.vercel.app/api/auth/password.
Reset' \
2 --header 'Content-Type: application/json' \
3 --header 'Authorization: *****' \
4 --data-raw '{
5 "email": "s4084220@student.zmit.edu.au",
6 "action": "request-reset"
7 }'

1.5 Reset Password (ValidateToken)

API URL	POST {{BASE_URL}}/api/auth/password.Reset
Request	<pre>{ "token": "string", "newPassword": "string", "email": "string", "action": "string" }</pre>
Response	<pre>{ "success": true, "message": "Password reset successfully", "statusCode": "HttpStatus" }</pre>
Example	<pre>curl --location 'http://nodejs-serverless-function-express-rho-ashen.vercel.app/api/auth/password.Reset' \ --header 'Content-Type: application/json' \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxNCIsImVtYWlsIjoienp6ekBkLmNvbSIsInVzZXJJZCI6MTQsImth dCI6MTc1ODYyNTU5OSwiZXhwIjoxNzU4NzExOTk5fQ.elaS_DVEgAiSPOPvc5GgcXaNaYjD37u-jSBq9DzgrEs' \ --data-raw ' { "email": "s4084228@student.rmit.edu.au", "action": "verify-token", "token": "8D8BDAAD", "newPassword": "NewSecurePass123" }'</pre>

```
{
  "success": true,
  "message": "Password reset successful",
  "data": {},
  "statusCode": 200
}
```

```
{
  "success": false,
  "message": "Invalid or expired reset code",
  "statusCode": 400
}
```



1.6 (Optional) Refresh Token

API URL	POST {{BASE_URL}}/api/auth/refresh
Request	{ "refreshToken": "string" }
Response	{ "success": true, "data": { "accessToken": "string", "refreshToken": "string" } }
Example	

2. Theory of Change Management (NoSQL)

2.1 Create Project Name

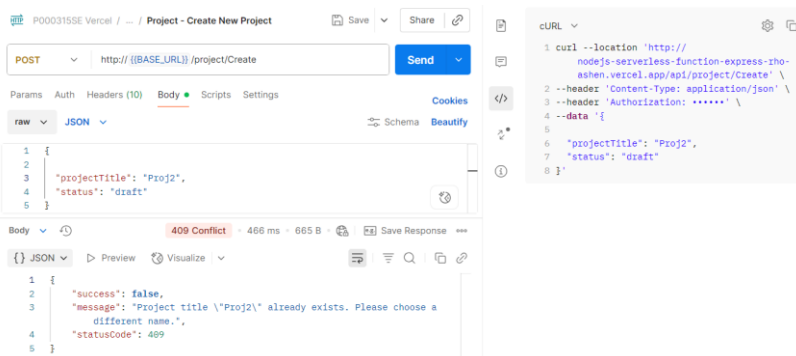
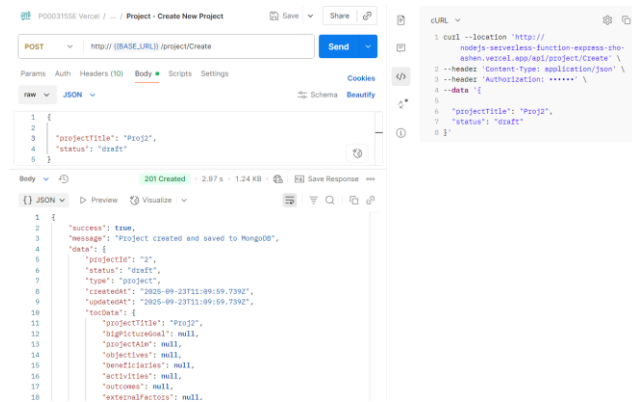
API URL	POST {{BASE_URL}}/api/project/Create
Request	<pre>{ "userId": "string", "projectTitle": "string", "status": "string" }</pre>
Response	<pre>{ "success": boolean, "message": "string", "data": {ProjectData } }, "statusCode": "HttpStatus" }</pre>
Example	<pre>curl --location 'http://nodejs-serverless-function-express-rho-ashen.vercel.app/api/project/Create' \ --header 'Content-Type: application/json' \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiNlbnVtYXVzZXJZCjZCMTQsImhhdCI6MTc1ODYyNTU5OSwiZXhwIjoxNzU4NzExOTk5fQ.elaS_DVEgAiSPOPvc5GgcXaNaYjD37u-jSBq9DzgrEs' \ --data '{ "projectTitle": "Proj2", "status": "draft" }' { "success": true, "message": "Project created and saved to MongoDB", "data": { "projectId": "2", "status": "draft", "type": "project", "createdAt": "2025-09-23T11:09:59.739Z", "updatedAt": "2025-09-23T11:09:59.739Z", "tocData": { "projectTitle": "Proj2", "bigPictureGoal": null, "projectAim": null, "objectives": null, "beneficiaries": null, "activities": null, "outcomes": null, "externalFactors": null, "evidenceLinks": null }, "tocColor": { "bigPictureGoal": { "shape": "", "text": "" }, "projectAim": { "shape": "", "text": "" }, "activities": {</pre>

```

    "shape": "",
    "text": ""
  },
  "objectives": {
    "shape": "",
    "text": ""
  },
  "beneficiaries": {
    "shape": "",
    "text": ""
  },
  "outcomes": {
    "shape": "",
    "text": ""
  },
  "externalFactors": {
    "shape": "",
    "text": ""
  },
  "evidenceLinks": {
    "shape": "",
    "text": ""
  }
},
"statusCode": 201
}

{
  "success": false,
  "message": "Project title \"Proj2\" already exists. Please choose a different name.",
  "statusCode": 409
}

```



2.2 Rename Project

API URL	PUT {{BASE_URL}}/api/project/Update
Request	<pre>{ "projectId": "string", "projectTitle": "string", "updateName": Boolean, "status": "string" }</pre>
Response	<pre>{ "success": boolean, "message": "String", "data": { ProjectData }, "statusCode": "Http.Status" }</pre>
Example	<pre>curl --location --request PUT 'http://nodejs-serverless-function-express-rho-ashen.vercel.app/api/project/Update' \ --header 'Content-Type: application/json' \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWUiOiJNCiIsImVtYWlsjoiWVtYWlsjoienp6ekBkLmNvbSIsInVzZXJJZCI6MTQsImhhdCI6MTc1O0DYyNTU5OSwiZXhwIjoxNzU4NzExOTk5fQ.elaS_DVEgAiSPOPvc5GgcXaNaYjD37u-jSBq9DzgrEs' \ --data '{ "projectId": "1", "projectTitle": "Proj1_z", "updateName": true, "status": "draft" }'</pre> <pre>{ "success": true, "message": "Project updated successfully", "data": { "projectId": "1", "status": "draft", "type": "project", "createdAt": "2025-09-23T11:08:06.640Z", "updatedAt": "2025-09-23T11:08:43.673Z", "tocData": { "projectTitle": "Proj1_z", "bigPictureGoal": null, "projectAim": null, "objectives": null, "beneficiaries": null, "activities": null, "outcomes": null, "externalFactors": null, "evidenceLinks": null }, "tocColor": { "bigPictureGoal": { "shape": "", "text": "" }, "projectAim": { "shape": "", "text": "" }, "activities": { </pre>

```
    "shape": "",
    "text": ""
  },
  "objectives": {
    "shape": "",
    "text": ""
  },
  "beneficiaries": {
    "shape": "",
    "text": ""
  },
  "outcomes": {
    "shape": "",
    "text": ""
  },
  "externalFactors": {
    "shape": "",
    "text": ""
  },
  "evidenceLinks": {
    "shape": "",
    "text": ""
  }
},
"statusCode": 200
}

{
  "success": false,
  "message": "Project 10 not found for user 14",
  "statusCode": 404
}
```

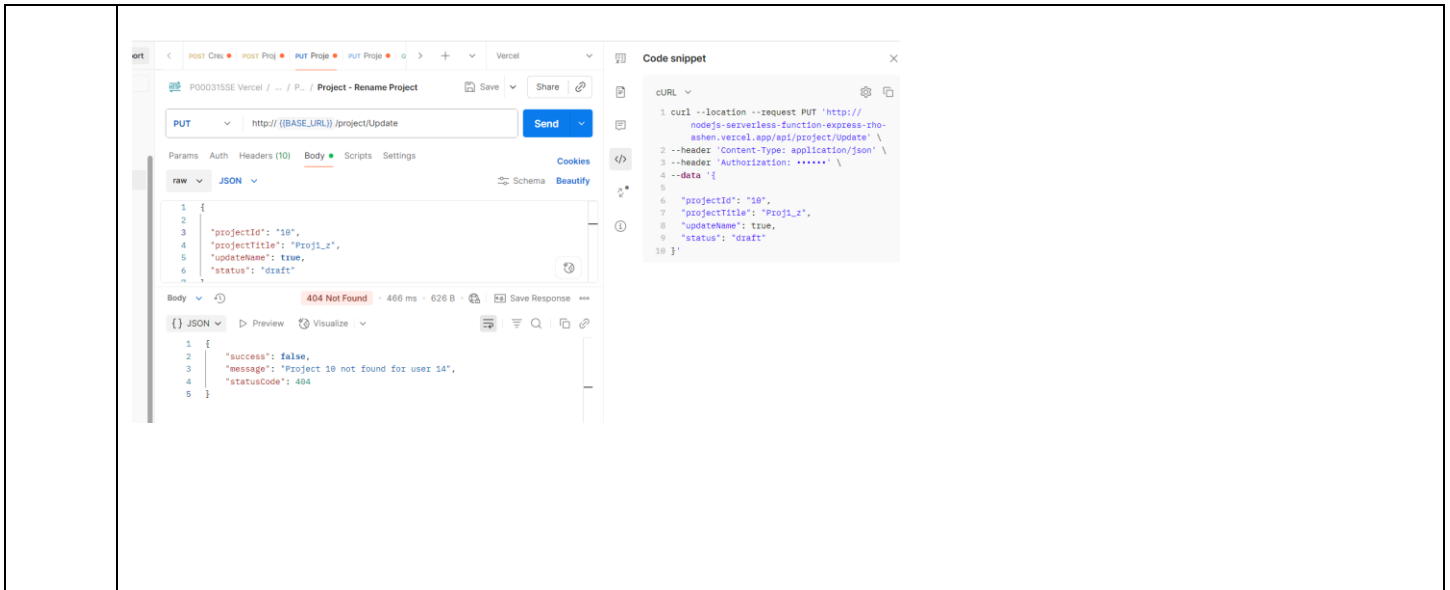
The screenshot displays a REST client interface with a PUT request and its response. The request is sent to `http://([BASE_URL])/project/update` with a JSON body. The response is a 200 OK status with a JSON body indicating success.

Request:

```
PUT http://([BASE_URL])/project/update
Content-Type: application/json
Authorization: *****
{
  "projectId": "1",
  "projectTitle": "Proj1_2",
  "updateName": true,
  "status": "draft"
}
```

Response:

```
200 OK
{
  "success": true,
  "message": "Project updated successfully",
  "data": {
    "projectId": "1",
    "status": "draft",
    "type": "project",
    "createdAt": "2025-09-23T11:08:06.648Z",
    "updatedAt": "2025-09-23T11:08:43.673Z",
    "toData": {
      "projectTitle": "Proj1_2",
      "bigPictureGoal": null,
      "projectAim": null,
      "objectives": null,
      "beneficiaries": null,
      "activities": null,
      "outcomes": null
    }
  }
}
```



2.3 Save TOC (Update TOC content in existing project)

API URL	PUT {{BASE_URL}}/api/project/Update
Request	<pre>{ "userId": "string", "projectId": "string", "projectTitle": "string", "updateName": Boolean, "status": "draft published", "tocData": { }, "tocColor": { } }</pre>
Response	<pre>{ "success": "boolean", "message": "string", "data": { "userId": "string", "projectId": "string", "tocData": { }, "tocColor": { }, "status": "draft published", "type": "string", "updatedAt": "string", "createdAT": "string" }, "statusCode": "HTTP.Status" }</pre>
Example	<pre>curl --location --request PUT 'http://nodejs-serverless-function-express-rho-ashen.vercel.app/api/project/Update' \ --header 'Content-Type: application/json' \</pre>


```
--header 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWUiOiNClmVtYWlsIjoienp6ekBkLmNvbSIsInVzZXJJZCI6MTQsImhh
dCI6MTc1ODYyNTU5OSwiZXhwIjoxNzU4NzExOTk5fQ.elaS_DVEgAiSPOpvc5GgcXaNaYjD37u-jSBq9DzgrEs' \
--data '{

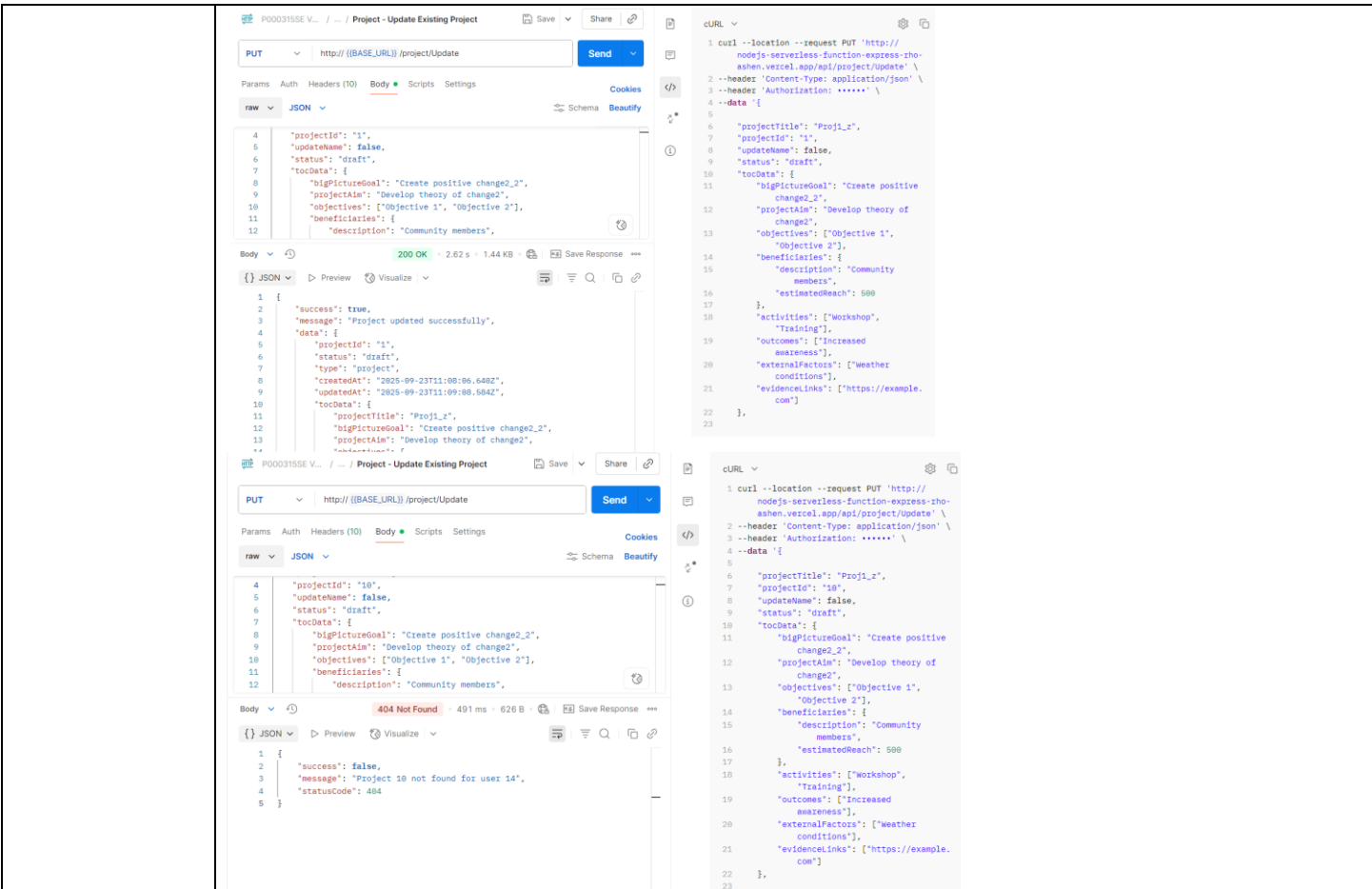
  "projectTitle": "Proj1_z",
  "projectId": "1",
  "updateName": false,
  "status": "draft",
  "tocData": {
    "bigPictureGoal": "Create positive change2_2",
    "projectAim": "Develop theory of change2",
    "objectives": ["Objective 1", "Objective 2"],
    "beneficiaries": {
      "description": "Community members",
      "estimatedReach": 500
    },
    "activities": ["Workshop", "Training"],
    "outcomes": ["Increased awareness"],
    "externalFactors": ["Weather conditions"],
    "evidenceLinks": ["https://example.com"]
  },

  "tocColor": {
    "bigPictureGoal": {
      "shape": "black",
      "text": "yellow"
    },
    "projectAim": {
      "shape": "",
      "text": ""
    },
    "activities": {
      "shape": "",
      "text": ""
    },
    "objectives": {
      "shape": "",
      "text": ""
    }
  }
}'

{
  "success": true,
  "message": "Project updated successfully",
  "data": {
    "projectId": "1",
    "status": "draft",
    "type": "project",
    "createdAt": "2025-09-23T11:08:06.640Z",
    "updatedAt": "2025-09-23T11:09:08.584Z",
    "tocData": {
      "projectTitle": "Proj1_z",
      "bigPictureGoal": "Create positive change2_2",
      "projectAim": "Develop theory of change2",
      "objectives": [
        "Objective 1",
        "Objective 2"
      ],
      "beneficiaries": {
        "description": "Community members",
        "estimatedReach": 500
      }
    },
  },
}
```

```
"activities": [
  "Workshop",
  "Training"
],
"outcomes": [
  "Increased awareness"
],
"externalFactors": [
  "Weather conditions"
],
"evidenceLinks": [
  "https://example.com"
]
},
"tocColor": {
  "bigPictureGoal": {
    "shape": "black",
    "text": "yellow"
  },
  "projectAim": {
    "shape": "",
    "text": ""
  },
  "activities": {
    "shape": "",
    "text": ""
  },
  "objectives": {
    "shape": "",
    "text": ""
  },
  "beneficiaries": {
    "shape": "",
    "text": ""
  },
  "outcomes": {
    "shape": "",
    "text": ""
  },
  "externalFactors": {
    "shape": "",
    "text": ""
  },
  "evidenceLinks": {
    "shape": "",
    "text": ""
  }
}
},
"statusCode": 200
}

{
  "success": false,
  "message": "Project 10 not found for user 14",
  "statusCode": 404
}
```



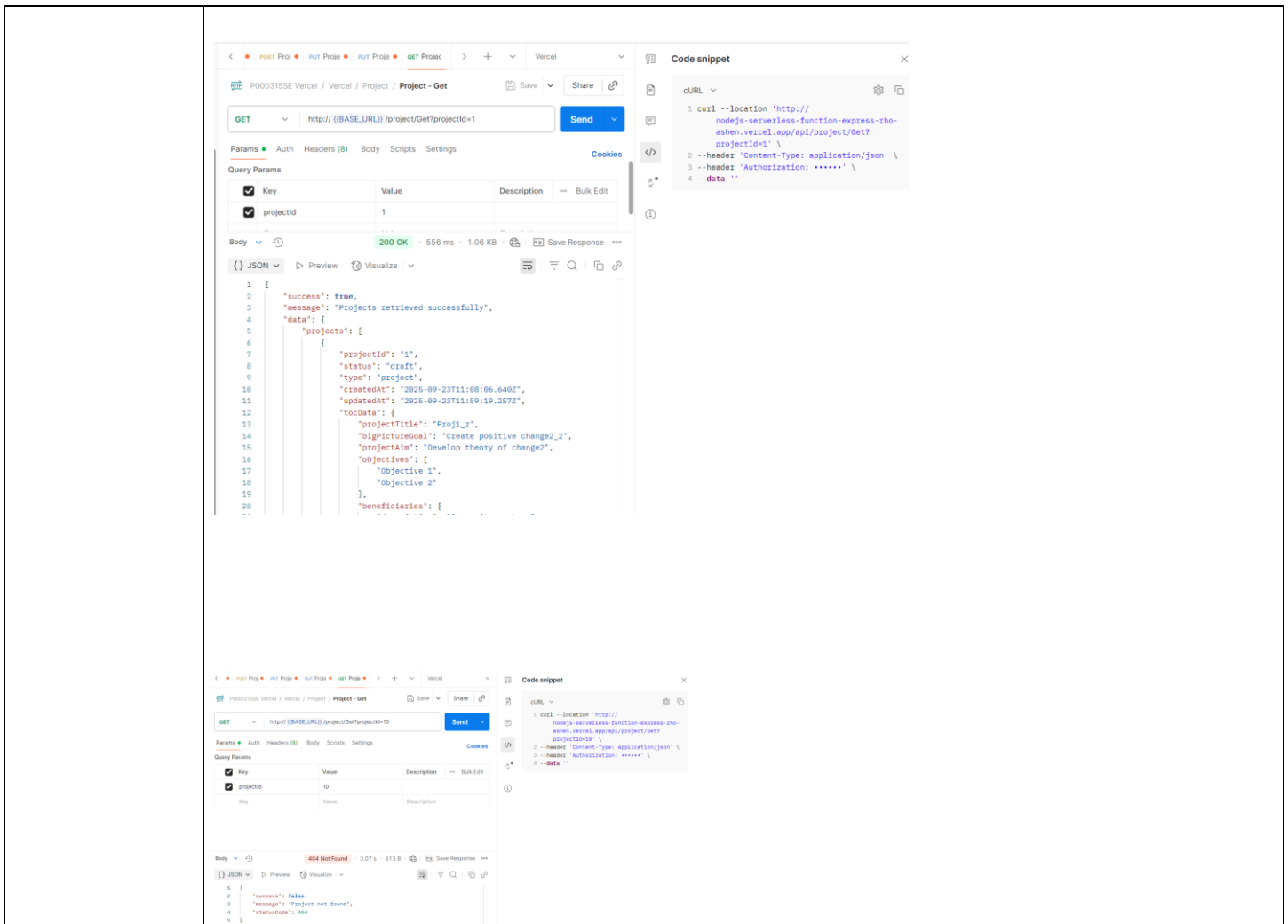
2.2 Get TOC

API URL	GET {{BASE_URL}}/api/project/Get
Request	'http://localhost:3000/api/project/toc.Get?projectId="string"'
Response	{ "success": "boolean", "message": "string", "data": { "projects": [{ "userId": "string", "projectId": "string", "tocData": { }, "tocColor": { }, "status": "draft published", "type": "string", "createdAt": "Datetime", "updatedAt": "DateTime" }], "pagination": { "page": "number",

	<pre>"limit": "number", "total": "number", "totalPages": "number" } }, "statusCode": "HTTP.Status" }</pre>
Example	<pre>curl --location 'http://nodejs-serverless-function-express-rho-ashen.vercel.app/api/project/Get?projectId=1' \ --header 'Content-Type: application/json' \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiNlbnVtYWlsIjoienp6ekBkLmNvbSIsInVzZXJJZCI6MTQsImIh dCI6MTc1ODYyNTU5OSwiZXhwIjojNzU4NzExOTk5fQ.elaS_DVEgAiSPOPvc5GgcXaNaYjD37u-jSBq9DzgrEs' \ --data "</pre> <pre>{ "success": true, "message": "Projects retrieved successfully", "data": { "projects": [{ "projectId": "1", "status": "draft", "type": "project", "createdAt": "2025-09-23T11:08:06.640Z", "updatedAt": "2025-09-23T11:09:08.584Z", "tocData": { "projectTitle": "Proj1_z", "bigPictureGoal": "Create positive change2_2", "projectAim": "Develop theory of change2", "objectives": ["Objective 1", "Objective 2"], "beneficiaries": { "description": "Community members", "estimatedReach": 500 }, "activities": ["Workshop", "Training"], "outcomes": ["Increased awareness"], "externalFactors": ["Weather conditions"], "evidenceLinks": ["https://example.com"] }, "tocColor": { "bigPictureGoal": { "shape": "black", "text": "yellow" }, "projectAim": { "shape": "", "text": "" }, "activities": {</pre>

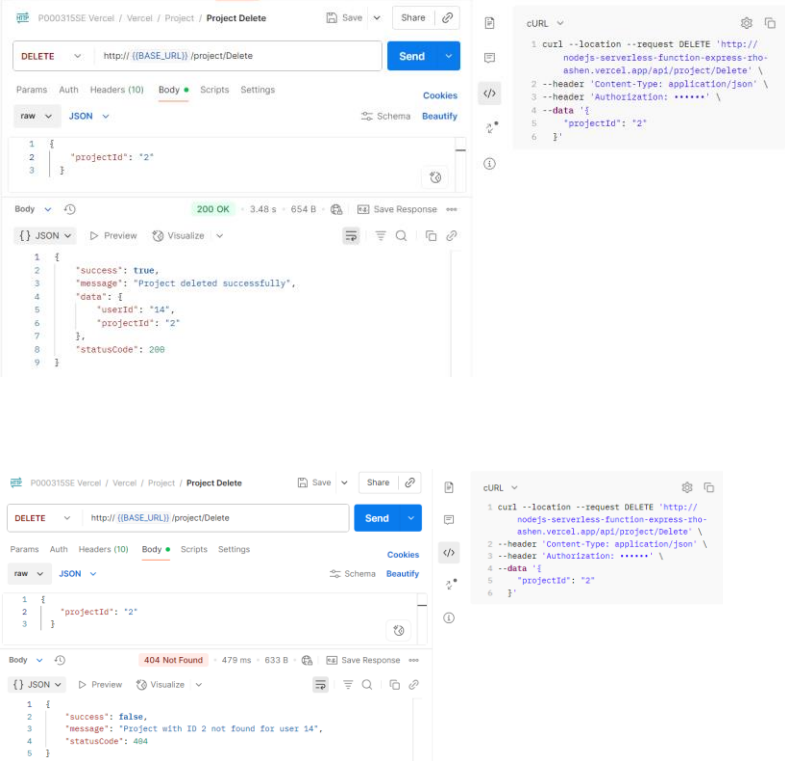
```
        "shape": "",
        "text": ""
      },
      "objectives": {
        "shape": "",
        "text": ""
      },
      "beneficiaries": {
        "shape": "",
        "text": ""
      },
      "outcomes": {
        "shape": "",
        "text": ""
      },
      "externalFactors": {
        "shape": "",
        "text": ""
      },
      "evidenceLinks": {
        "shape": "",
        "text": ""
      }
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 10,
    "total": 1,
    "totalPages": 1
  }
},
"statusCode": 200
}

{
  "success": false,
  "message": "Project not found",
  "statusCode": 404
}
```



2.3 (Optional) Delete TOC

API URL	DELETE {{BASE_URL}}/api/project/Delete
Request	{ "projectId": "string" }
Response	{ "success": boolean, "message": "string", "data": { }, "statusCode": HTTPS.Status }
Example	<pre>curl --location --request DELETE 'http://nodejs-serverless-function-express-rho-ashen.vercel.app/api/project/Delete' \ --header 'Content-Type: application/json' \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJlNCIsImVtYWlsIjoienp6ekBkLmNvbSIsInVzZXJJZCI6MTQsImthdCI6MTc1ODYyNTU0S0wiZXhwIjoxNzU4NzExOTk5fQ.elaS_DVEgAiSPOPvc5GgcXaNaYjD37u-jSBq9DzgrEs' \ --data '{ "projectId": "2" }'</pre>

	<pre>"success": true, "message": "Project deleted successfully", "data": { "userId": "14", "projectId": "2" }, "statusCode": 200 } { "success": false, "message": "Project with ID 2 not found for user 14", "statusCode": 404 }</pre>  <p>The image contains two screenshots of a REST client interface. The top screenshot shows a successful DELETE request to 'http://{{BASE_URL}}/project/Delete' with a body of {'projectId': '2'}, resulting in a 200 OK response with a JSON body: {'success': true, 'message': 'Project deleted successfully', 'data': {'userId': '14', 'projectId': '2'}, 'statusCode': 200}. The bottom screenshot shows the same request but with a 404 Not Found response, with a JSON body: {'success': false, 'message': 'Project with ID 2 not found for user 14', 'statusCode': 404}. To the right of each screenshot is a cURL command: 'curl --location --request DELETE 'http://nodejs-serverless-function-express-rho-ashen.vercel.app/api/project/Delete' \ --header 'Content-Type: application/json' \ --header 'Authorization: *****' \ --data '{ "projectId": "2" }'</p>
--	---

2.4 (Optional) List user TOCs

API URL	GET {{BASE_URL}}/api/project/GetProjectList
Request	
Response	{ "success": boolean, "message": "string", "data": { "projects": [{ProjectData},{ProjectData}] }, "statusCode": HTTPS.Status }
Example	curl --location 'http://nodejs-serverless-function-express-rho-ashen.vercel.app/api/project/GetProjectList' \

```
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJmVtYWVlIjoienp6ekBkLmNvbSIsInVzZXJJZCI6MTQsImh
dCI6MTc1ODYyNTU5OSwiZXhwIjoxNzU4NzExOTk5fQ.elaS_DVEgAiSPOpvc5GgcXaNaYjD37u-jSBq9DzgrEs' \
--data "

{
  "success": true,
  "message": "Project list retrieved successfully",
  "data": {
    "projects": [
      {
        "projectId": "1",
        "projectName": "Proj1_z"
      },
      {
        "projectId": "2",
        "projectName": "Proj2"
      }
    ],
    "count": 2
  },
  "statusCode": 200
}

{
  "success": true,
  "message": "Project list retrieved successfully",
  "data": {
    "projects": [],
    "count": 0
  },
  "statusCode": 200
}

{
  "success": false,
  "message": "Project list retrieved unsuccessful",
  "statusCode": 400
}
```

P000315SE Vercel / ... / Pr... / Project - GetProjectList

GET

http://{{BASE_URL}}/project/GetProjectList

Send

Params

Auth

Headers (8)

Body

Scripts

Settings

Query Params

Body

200 OK

2.42 s

738 B

Save Response

JSON

Preview

Visualize

```
1 {
2   "success": true,
3   "message": "Project list retrieved successfully",
4   "data": {
5     "projects": [
6       {
7         "projectId": "1",
8         "projectName": "Proj1_z"
9       },
10      {
11        "projectId": "2",
12        "projectName": "Proj2"
13      }
14    ],
15    "count": 2
16  },
17  "statusCode": 200
18 }
```

cURL

1 curl --location 'http://
2 nodejs-serverless-function-express-zho-
3 ashen.vercel.app/api/project/
4 GetProjectList' \
5 --header 'Content-Type: application/json' \
6 --header 'Authorization: ' \
7 --data ''

3. Payment & Subscription Management

3.1 Create Payment

API URL	POST /api/payment/create-intent
Request	<pre>{ "user_id": "string", "plan_id": "string", "billing_interval": "string", "auto_renew": "boolean", "currency": "USD" }</pre>
Response	<pre>{ "success": true, "data": { "clientSecret": "string", "subscriptionId": "string", "amount": "number" } }</pre>
Example	

3.2 Confirm Payment

API URL	POST /api/payment/confirm
Request	<pre>{ "paymentIntentId": "string", "paymentMethodId": "string" }</pre>
Response	<pre>{ "success": true, "data": { "subscriptionId": "string", "status": "active", "expiresAt": "datetime" } }</pre>
Example	

3.3 Get Subscription Status

API URL	GET /api/subscription/status
Request	<pre>{</pre>

	<pre> "success": true, "data": { "plan": "free premium", "status": "active expired cancelled", "expiresAt": "datetime", "autoRenew": "boolean" } } </pre>
Response	
Example	

3.4 Cancel Subscription

API URL	POST /api/subscription/cancel
Request	
Response	<pre> { "success": true, "message": "Subscription cancelled successfully" } </pre>
Example	

3.5 (Need?) Update Subscription

Invoice Data Object Needs with APIs'

	<pre> { "invoice_id": "inv_12345abc", "subscription_id": "sub_67890def", "user_id": "user_12345", "amount_cents": 2999, "currency": "USD", "period_start": "2025-08-01", "period_end": "2025-08-31", "issued_at": "2025-08-27T10:00:00Z", "due_at": "2025-09-10T23:59:59Z", "status": "pending", "pdf_url": "https://invoices.example.com/inv_12345abc.pdf", "is_public": false } </pre>

4. Admin Dashboard

4.1 Get Subscription User Details

API URL	
Request	
Response	
Example	

4.2 Update Subscription Details

API URL	
Request	
Response	
Example	

4.3 Get Registered User Details

API URL	
Request	
Response	
Example	

4.4 (Need?) Get Revenue

API URL	
Request	
Response	
Example	

Need Plan Create/Update/Delete APIs

	<pre>{ "plan_id": "string", "name": "string", "price_cents": "number", "billing_interval": "monthly yearly weekly quarterly" }</pre>

5. (Optional) Export & File Management

5.1 Export TOC as PDF

API URL	POST /api/export/pdf
Request	<pre>{ "tocId": "string", "visualizationData": "string", // SVG or Canvas data from frontend "format": { "orientation": "portrait landscape", "size": "A4 A3 letter" } }</pre>
Response	<pre>{ "success": true, "data": { "fileId": "string", "downloadUrl": "string", "filename": "string", "expiresAt": "datetime" } }</pre>
Example	

5.2 Export TOC as PNG

API URL	POST /api/export/png
Request	<pre>{ "tocId": "string", "visualizationData": "string", "resolution": { "width": "number", "height": "number", "dpi": "number" } }</pre>
Response	
Example	

5.3 Get Download URL

API URL	GET /api/exports?tocId={tocId}
Request	
Response	<pre>{ "success": true, "data": [{ "fileId": "string", </pre>

	<pre> "filename": "string", "format": "pdf png", "downloadUrl": "string", "createdAt": "datetime", "expiresAt": "datetime" }] } </pre>
Example	

6. (Optional) Help Guide & Content Management

6.1 Get Help Content

API URL	GET /api/help/{section}
Request	
Response	<pre> { "success": true, "data": { "section": "string", "title": "string", "content": "string", "lastUpdated": "datetime" } } </pre>
Example	

6.2 Get Help Section

API URL	GET /api/help/sections
Request	
Response	<pre> { "success": true, "data": [{ "id": "string", "title": "string", "description": "string", "order": "number" }] } </pre>
Example	

7. (Optional) Session Management

7.1 Save Session Data

API URL	POST /api/session
Request	{ "data": "object", // Contains userId internally "expiresAt": "datetime" }
Response	
Example	

7.2 Get Session Data

API URL	GET /api/session/{sessionId}
Request	
Response	
Example	

8. (Optional) Email Service

8.1 Send Email (Internal Use)

API URL	POST /api/email/send
Request	{ "to": "string", "template": "welcome password_reset payment_confirmation", "data": "object" }
Response	
Example	

9. (Optional) Health & Monitoring

9.1 Health Check

API URL	GET /api/health
Request	
Response	
Example	

Parameter Details

Field Name	Data Type	Min Length	Max Length	Pattern/Format	Required	Allowed Values	Example	Additional Constraints

email	string	5	254	RFC 5322 email format	Yes	Valid email format	"user@example.com"	Must be unique, case-insensitive
password	string	8	128	Mixed case, numbers, special chars	Yes	a-zA-Z0-9!@#\$\$%^&*()_+ -=	"MyPass123!"	Must contain: 1 uppercase, 1 lowercase, 1 number, 1 special char
firstName	string	1	50	Letters, spaces, hyphens, apostrophes	Yes	a-zA-Z\s-	"John"	No leading/trailing spaces
lastName	string	1	50	Letters, spaces, hyphens, apostrophes	Yes	a-zA-Z\s-	"Doe-Smith"	No leading/trailing spaces
organization	string	1	100	Alphanumeric, spaces, common punctuation	No	a-zA-Z0-9\s-.,&()	"ACME Corp. & Associates"	Optional field
userId	string	24	24	MongoDB ObjectId format	System	Hex characters	"507f1f77bcf86cd799439011"	Auto-generated, immutable
accessToken	string	200	500	JWT format	System	Base64 URL safe	"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."	Auto-generated, expires in 1 hour
refreshToken	string	200	500	JWT format	System	Base64 URL safe	"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."	Auto-generated, expires in 30 days
token	string	32	64	Hexadecimal	System	0-9a-fA-F	"a1b2c3d4e5f6789012345678901234ab"	Password reset token, expires in 1 hour
newPassword	string	8	128	Mixed case, numbers, special chars	Yes	a-zA-Z0-9!@#\$\$%^&*()_+ -=	"NewPass456@"	Same as password constraints
projectTitle	string	3	200	Alphanumeric, spaces, punctuation	Yes	UTF-8 characters	"Clean Water Initiative 2024"	Must be unique per user
bigPictureGoal	string	10	1000	Text with basic formatting	Yes	UTF-8 characters	"Improve access to clean water in rural communities"	Rich text allowed

projectAim	string	10	1000	Text with basic formatting	Yes	UTF-8 characters	"Reduce waterborne diseases by 50% in target areas"	Rich text allowed
objectives	array[string]	1 item	10 items	Each string 5-500 chars	Yes	UTF-8 text array	["Install 20 water pumps", "Train 100 community members"]	Each objective must be unique
beneficiaries.description	string	10	500	Text description	Yes	UTF-8 characters	"Rural families in 5 villages"	Descriptive text
beneficiaries.estimatedReach	number	1	999999999	Positive integer	Yes	Whole numbers	1500	Must be realistic estimate
activities	array[string]	1 item	20 items	Each string 5-300 chars	Yes	UTF-8 text array	["Drill wells", "Install pumps", "Conduct training"]	Each activity must be unique
outcomes	array[string]	1 item	15 items	Each string 5-400 chars	Yes	UTF-8 text array	["50% reduction in waterborne illness", "Improved school attendance"]	Measurable outcomes preferred
externalFactors	array[string]	0 items	10 items	Each string 5-300 chars	No	UTF-8 text array	["Weather conditions", "Government policy changes"]	Risk factors or assumptions
evidenceLinks	array[string]	0 items	20 items	Valid URL format	No	HTTP/HTTPS URLs	["https://example.com/research.pdf"]	Must be accessible URLs
status	string	4	9	Predefined values	Yes	"draft", "published"	"draft"	Controls visibility
objectId	string	24	24	MongoDB ObjectId format	System	Hex characters	"507f1f77bcf86cd799439012"	Auto-generated, immutable
createdAt	datetime	-	-	ISO 8601 format	System	UTC datetime	"2024-08-24T10:30:00.000Z"	Auto-generated, immutable
updatedAt	datetime	-	-	ISO 8601 format	System	UTC datetime	"2024-08-24T15:45:30.000Z"	Auto-updated on changes
plan	string	4	20	Predefined subscription plans	Yes	"premium_monthly", "premium_annual"	"premium_monthly"	Business logic determines pricing
currency	string	3	3	ISO 4217 currency code	Yes	"USD", "EUR", "GBP"	"USD"	Must be supported currency
clientSecret	string	20	100	Payment processor format	System	Alphanumeric + underscore	"pi_1234567890_secret_abcdef"	Stripe payment intent secret
paymentIntentId	string	20	50	Payment processor ID	System	Alphanumeric + underscore	"pi_1234567890abcdef"	Stripe payment intent ID
paymentMethodId	string	20	50	Payment processor ID	Yes	Alphanumeric + underscore	"pm_1234567890abcdef"	Stripe payment method ID

subscriptionId	string	20	50	Payment processor ID	System	Alphanumeric + underscore	"sub_1234567890abcdef"	Stripe subscription ID
amount	number	0.01	99999.99	Decimal currency amount	System	Positive decimal	29.99	In smallest currency unit for processing
expiresAt	datetime	-	-	ISO 8601 format	System	Future UTC datetime	"2024-12-24T23:59:59.000Z"	Must be future date
autoRenew	boolean	-	-	Boolean value	System	true, false	TRUE	Subscription auto-renewal setting
visualizationData	string	100	5000000	SVG/Canvas data	Yes	Valid SVG/Canvas	"<svg>...</svg>"	Must be valid XML/JSON
format.orientation	string	8	9	Print orientation	No	"portrait", "landscape"	"portrait"	Default: portrait
format.size	string	2	6	Paper size	No	"A4", "A3", "letter"	"A4"	Default: A4
resolution.width	number	100	10000	Pixel width	Yes	Positive integer	1920	Max depends on plan
resolution.height	number	100	10000	Pixel height	Yes	Positive integer	1080	Max depends on plan
resolution.dpi	number	72	600	Dots per inch	No	Positive integer	300	Default: 300 for PNG
fileId	string	24	24	MongoDB ObjectId format	System	Hex characters	"507f1f77bcf86cd799439013"	Auto-generated file identifier
filename	string	5	100	Valid filename	System	Alphanumeric, dash, dot	"project-report-2024.pdf"	Generated based on project title
downloadUrl	string	50	500	Valid HTTPS URL	System	HTTPS URL	" https://api.example.com/downloads/abc123 "	Temporarily signed URL
section	string	3	50	Help section identifier	Yes	Lowercase, hyphens	"getting-started"	Must match existing sections
title	string	5	200	Help content title	System	UTF-8 characters	"Getting Started Guide"	Human-readable title
content	string	10	50000	Markdown content	System	Markdown format	"# Welcome\n\nThis guide..."	Rich markdown content
lastUpdated	datetime	-	-	ISO 8601 format	System	UTC datetime	"2024-08-24T12:00:00.000Z"	Content modification timestamp
description	string	10	500	Section description	System	UTF-8 characters	"Learn how to create your first project"	Brief section

								description
order	number	1	1000	Display order	System	Positive integer	10	Determines section ordering
data	object	-	-	JSON object	Yes	Valid JSON	{"tocId": "...", "step": 3}	Session-specific data
sessionId	string	24	24	MongoDB ObjectId format	System	Hex characters	"507f1f77bcf86cd799439014"	Auto-generated session ID
to	string	5	254	Email address	Yes	Valid email format	"user@example.com"	Recipient email address
template	string	5	50	Email template name	Yes	"welcome", "password_reset", "payment_confirmation"	"welcome"	Must match available templates
page	number	1	1000	Page number	No	Positive integer	1	For pagination, default: 1
limit	number	1	100	Items per page	No	Positive integer	20	For pagination, default: 20, max: 100