BINAR FSW WAVE 32

CHALLENGE 8 - AGNES SEPTILIA

API DOCUMENTATION & REACT CLIENT

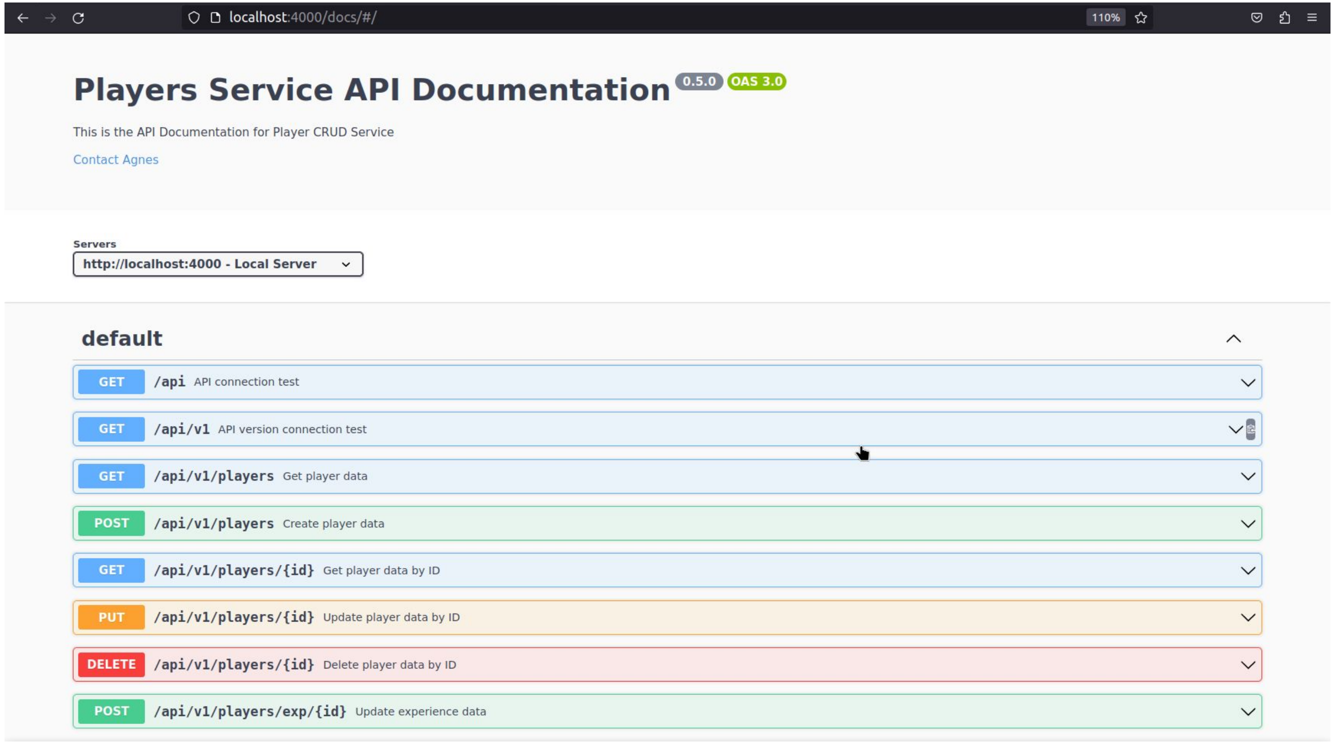
API DOCUMENTATION

PREPARATION

- API Documentation use Swagger in server side.
- SQL Script is available in the /*sql/init.sql*, Swagger script is available in /*routes/swagger-config.yaml.* Both can be found inside */server* folder
- Swagger runs in url **http://localhost:4000/docs/**

API DOCUMENTATION

Complete API List



localhost:4000/docs/#/

Players Service API Documentation 0.5.0 OAS 3.0

This is the API Documentation for Player CRUD Service

Contact Agnes

Servers

http://localhost:4000 - Local Server

default

GET `/api` API connection test

GET `/api/v1` API version connection test

GET `/api/v1/players` Get player data

POST `/api/v1/players` Create player data

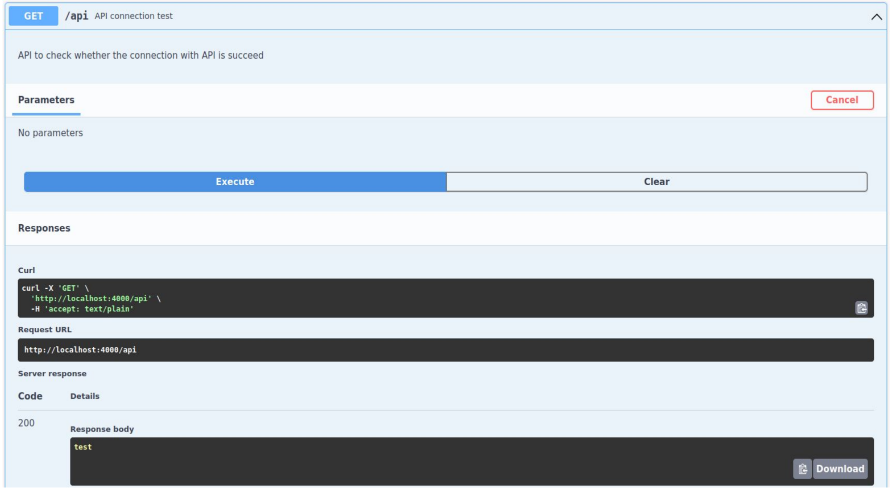
GET `/api/v1/players/{id}` Get player data by ID

PUT `/api/v1/players/{id}` Update player data by ID

DELETE `/api/v1/players/{id}` Delete player data by ID

POST `/api/v1/players/exp/{id}` Update experience data

API DOCUMENTATION



API connection test

API to check whether the connection with API is succeed

Parameters

No parameters

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:4000/api' \
  -H 'accept: text/plain'
```

Request URL

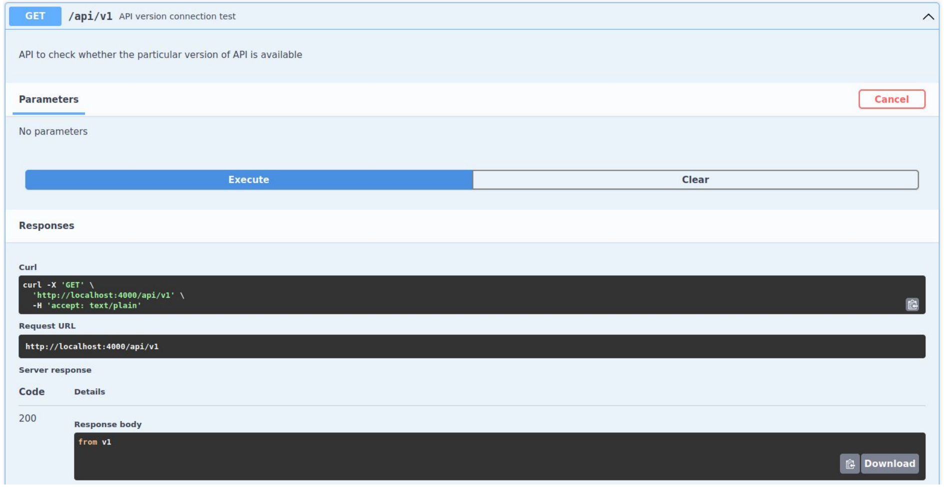
```
http://localhost:4000/api
```

Server response

Code | Details

200 — Response body

```
test
```



API version connection test

API to check whether the particular version of API is available

Parameters

No parameters

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:4000/api/v1' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:4000/api/v1
```

Server response

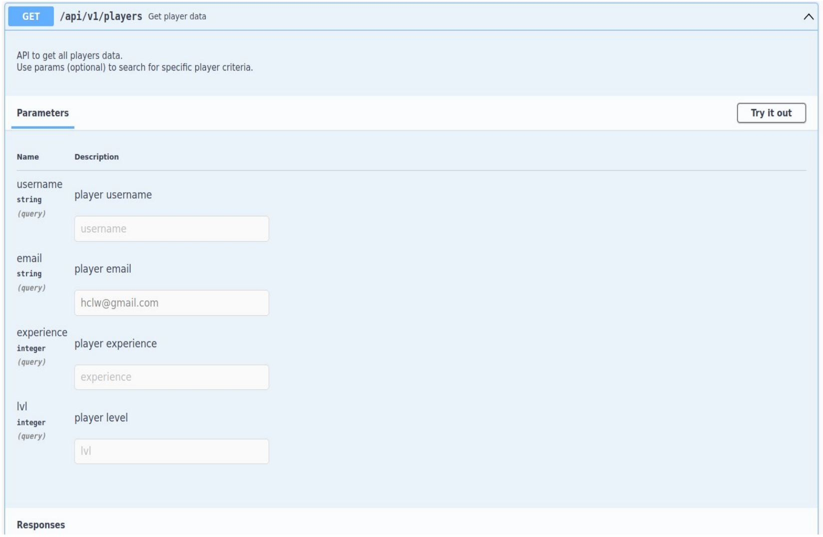
Code | Details

200 — Response body

```
from v1
```

The first two API is to test the connection to endpoint /api and /api/v1. Both using GET method.

API DOCUMENTATION


GET /api/v1/players Get player data

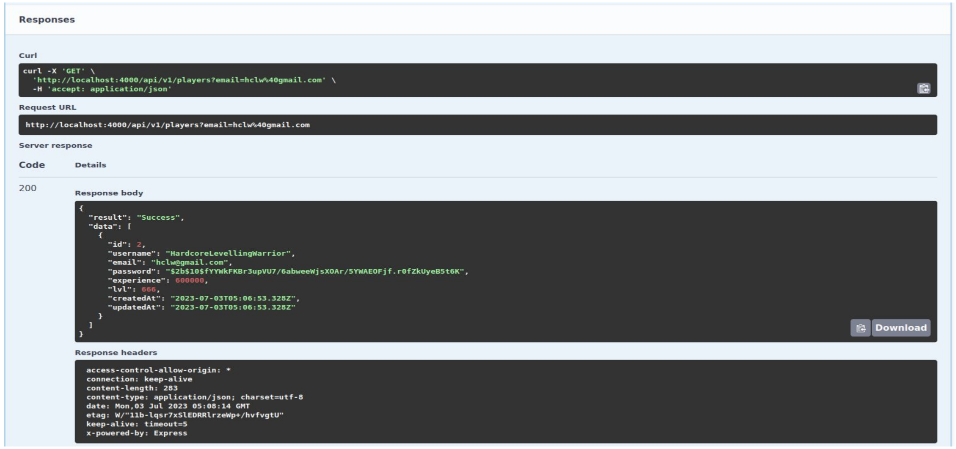
API to get all players data.
Use params (optional) to search for specific player criteria.

Parameters

Try it out

Name	Description
username string *(query)*	player username (username)
email string *(query)*	player email (hclw@gmail.com)
experience integer *(query)*	player experience (experience)
lvl integer *(query)*	player level (lvl)

Responses


Responses

Curl

```
curl -X 'GET' \
  'http://localhost:4000/api/v1/players?email=hclw%40gmail.com' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:4000/api/v1/players?email=hclw%40gmail.com
```

Server response

Code | Details

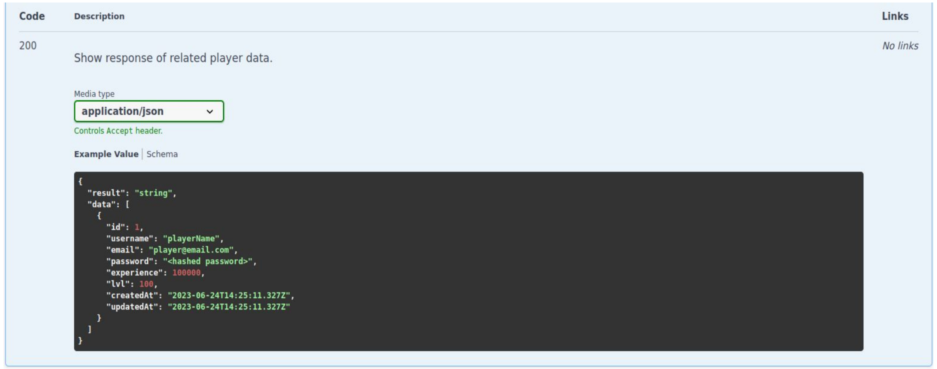
200 Response body

```
{
  "result": "Success",
  "data": [
    {
      "id": 2,
      "username": "HardcoreLevellingWarrior",
      "email": "hclw@gmail.com",
      "password": "$2b$10$fYYWkFKBr3upVU7/6abweeWjsXOAr/5YWAEOFjf.r0fZkUyeB5t6K",
      "experience": 600000,
      "lvl": 666,
      "createdAt": "2023-07-03T05:06:53.328Z",
      "updatedAt": "2023-07-03T05:06:53.328Z"
    }
  ]
}
```

Download

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 283
content-type: application/json; charset=utf-8
date: Mon,03 Jul 2023 05:08:14 GMT
etag: W/"11b-lqsr7xSlEDRRlrzeWp+/hvfvgtU"
keep-alive: timeout=5
x-powered-by: Express
```



Code	Description	Links
200	Show response of related player data.	*No links*

Media type

application/json

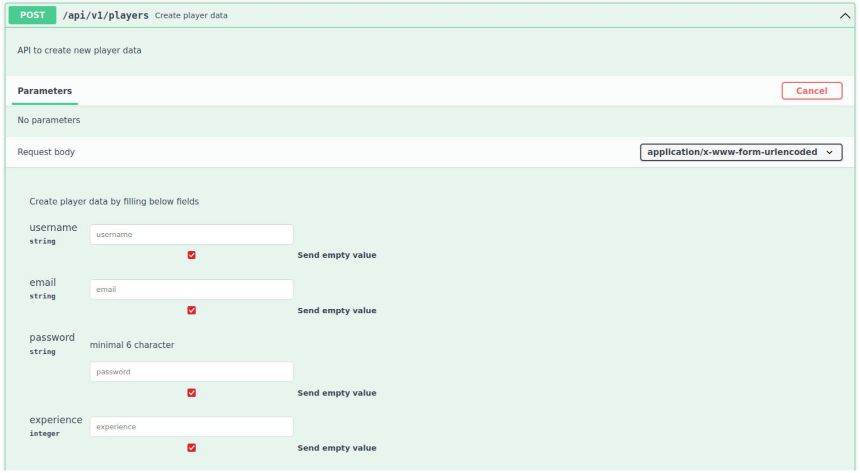
Controls Accept header.

Example Value | Schema

```
{
  "result": "string",
  "data": [
    {
      "id": 1,
      "username": "playerName",
      "email": "player@email.com",
      "password": "<hashed password>",
      "experience": 100000,
      "lvl": 100,
      "createdAt": "2023-06-24T14:25:11.327Z",
      "updatedAt": "2023-06-24T14:25:11.327Z"
    }
  ]
}
```

API to get player data
Endpoint: /api/v1/players
Method: GET
Parameters are optional.

API DOCUMENTATION



POST /api/v1/players Create player data

API to create new player data

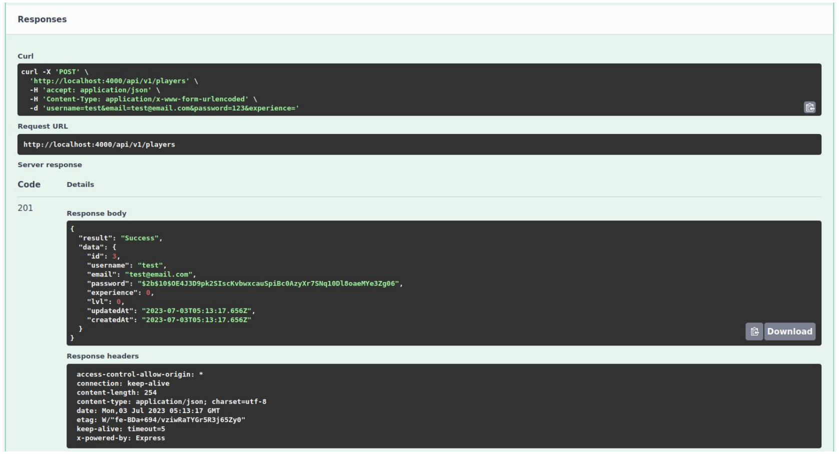
Parameters | Cancel

No parameters

Request body: application/x-www-form-urlencoded

Create player data by filling below fields

username *string*	username	☑ Send empty value
email *string*	email	☑ Send empty value
password *string*	minimal 6 character password	☑ Send empty value
experience *integer*	experience	☑ Send empty value



Responses

Curl

```
curl -X 'POST' \
  'http://localhost:4000/api/v1/players' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d 'username=test&email=test@email.com&password=123&experience='
```

Request URL

```
http://localhost:4000/api/v1/players
```

Server response

Code | Details

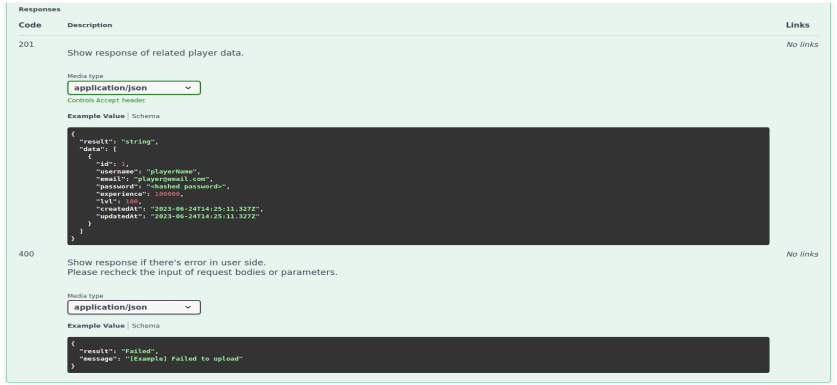
201

Response body

```
{
  "result": "Success",
  "data": {
    "id": 3,
    "username": "test",
    "email": "test@email.com",
    "password": "$2b$10$OE4J3D9pk2SIscKvbwxcauSpiBc0AzyXr7SNq10Dl8oaeMYe3Zg06",
    "experience": 0,
    "lvl": 0,
    "updatedAt": "2023-07-03T05:13:17.656Z",
    "createdAt": "2023-07-03T05:13:17.656Z"
  }
}
```

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 254
content-type: application/json; charset=utf-8
date: Mon,03 Jul 2023 05:13:17 GMT
etag: W/"fe-BDa+694/vziwRaTYGr5R3j65Zy0"
keep-alive: timeout=5
x-powered-by: Express
```



Responses

Code	Description	Links
201	Show response of related player data.	*No links*
400	Show response if there's error in user side. Please recheck the input of request bodies or parameters.	*No links*

201 — Media type: application/json. Controls Accept header.

Example Value | Schema

```
{
  "result": "string",
  "data": [
    {
      "id": 1,
      "username": "playerName",
      "email": "player@email.com",
      "password": "<hashed password>",
      "experience": 100000,
      "lvl": 100,
      "createdAt": "2023-06-24T14:25:11.327Z",
      "updatedAt": "2023-06-24T14:25:11.327Z"
    }
  ]
}
```

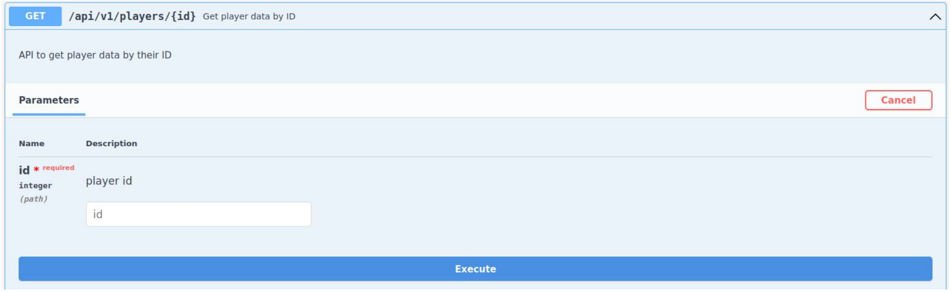
400 — Media type: application/json

Example Value | Schema

```
{
  "result": "Failed",
  "message": "[Example] Failed to upload"
}
```

API to create new player data
Endpoint: /api/v1/players
Method: POST
Username, email, and password (min. 6 character) are required.

API DOCUMENTATION

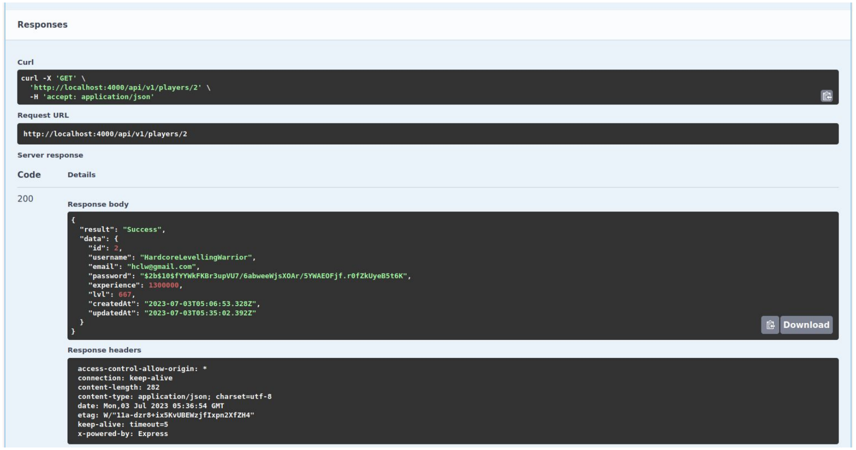


GET /api/v1/players/{id} Get player data by ID

API to get player data by their ID

Parameters

Name	Description
id * required integer *(path)*	player id id



Responses

Curl

```
curl -X 'GET' \
  'http://localhost:4000/api/v1/players/2' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:4000/api/v1/players/2
```

Server response

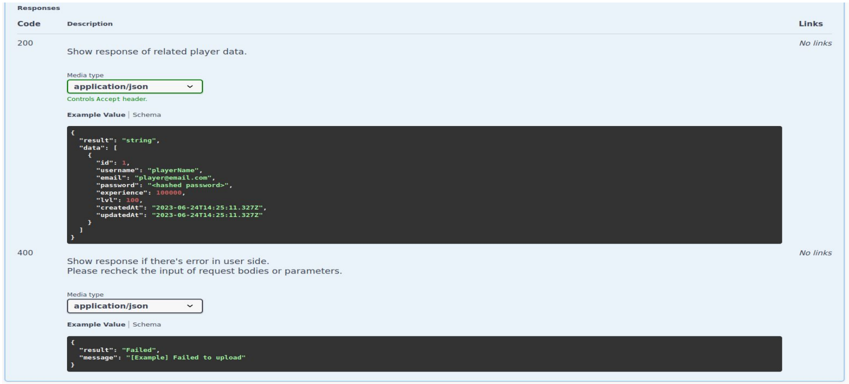
Code	Details
200	Response body

```
{
  "result": "Success",
  "data": {
    "id": 2,
    "username": "HardcoreLevellingWarrior",
    "email": "hclw@gmail.com",
    "password": "$2b$10$fYYWkFKBr3upVU7/6abweeWjsXOAr/5YWAEOFjf.r0fZkUyeB5t6K",
    "experience": 1300000,
    "lvl": 667,
    "createdAt": "2023-07-03T05:06:53.328Z",
    "updatedAt": "2023-07-03T05:35:02.392Z"
  }
}
```

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 282
content-type: application/json; charset=utf-8
date: Mon,03 Jul 2023 05:36:54 GMT
etag: W/"11a-dzr8+ix5KvUBEWzjfIxpn2XfZH4"
keep-alive: timeout=5
x-powered-by: Express
```

API to get player data by id
Endpoint: /api/v1/players/:id
Method: GET
ID is required.



Responses

Code	Description	Links
200	Show response of related player data.	*No links*
400	Show response if there's error in user side. Please recheck the input of request bodies or parameters.	*No links*

200 — Media type: application/json
Controls Accept header.

Example Value | Schema

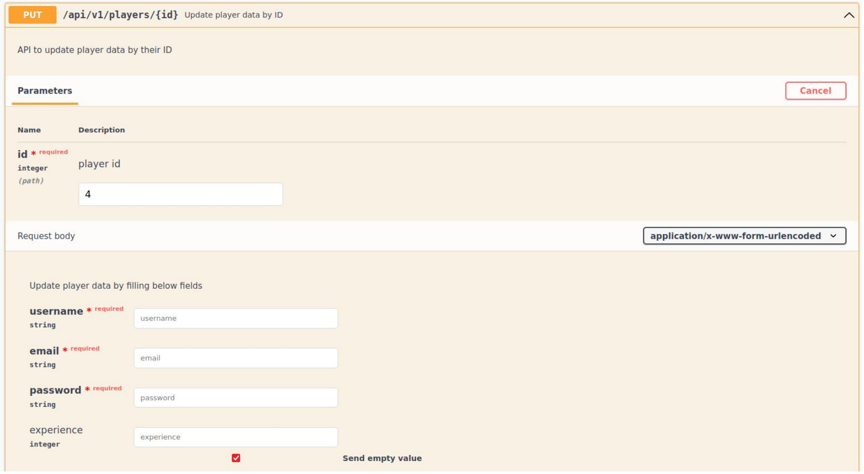
```
{
  "result": "string",
  "data": [
    {
      "id": 1,
      "username": "playerName",
      "email": "player@email.com",
      "password": "<hashed password>",
      "experience": 100000,
      "lvl": 100,
      "createdAt": "2023-06-24T14:25:11.327Z",
      "updatedAt": "2023-06-24T14:25:11.327Z"
    }
  ]
}
```

400 — Media type: application/json

Example Value | Schema

```
{
  "result": "Failed",
  "message": "[Example] Failed to upload"
}
```

API DOCUMENTATION



PUT `/api/v1/players/{id}` Update player data by ID

API to update player data by their ID

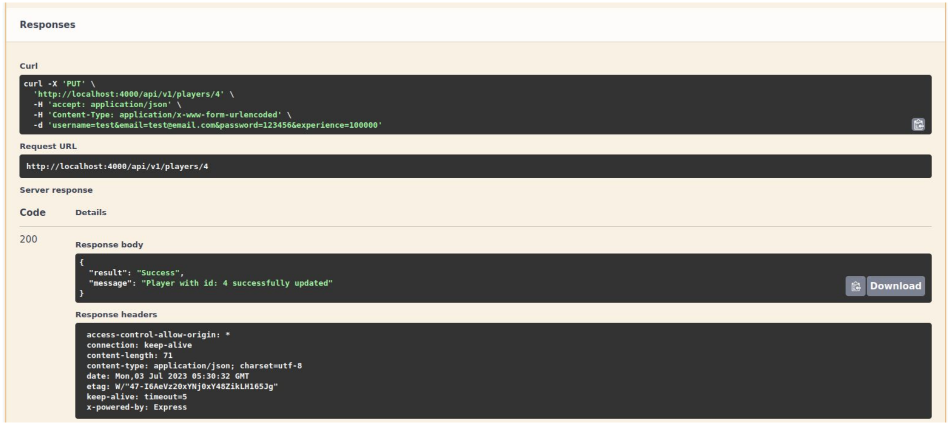
Parameters Cancel

Name	Description
id * required integer *(path)*	player id 4

Request body: application/x-www-form-urlencoded

Update player data by filling below fields

username * required string	username
email * required string	email
password * required string	password
experience integer	experience
✓	Send empty value



Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:4000/api/v1/players/4' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d 'username=test&email=test@email.com&password=123456&experience=100000'
```

Request URL

```
http://localhost:4000/api/v1/players/4
```

Server response

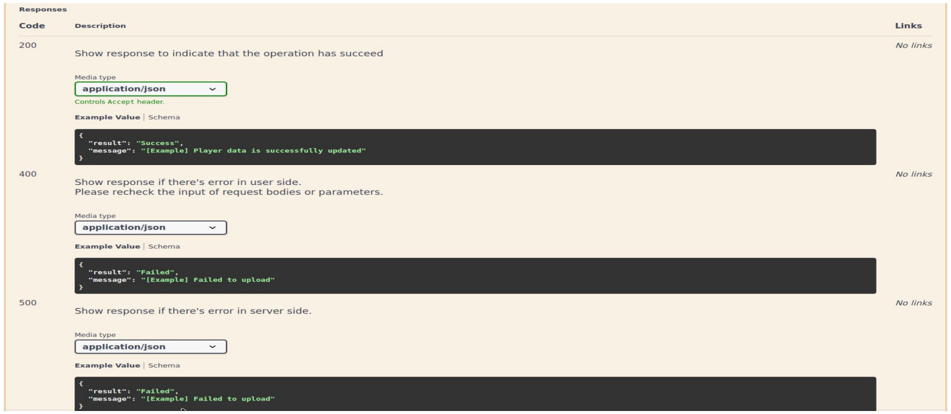
Code: 200

Response body

```
{
  "result": "Success",
  "message": "Player with id: 4 successfully updated"
}
```

Response headers

```
 access-control-allow-origin: *
 connection: keep-alive
 content-length: 71
 content-type: application/json; charset=utf-8
 date: Mon,03 Jul 2023 05:30:32 GMT
 etag: W/"47-I6AeVz20xYNj0xY48ZikLH165Jg"
 keep-alive: timeout=5
 x-powered-by: Express
```

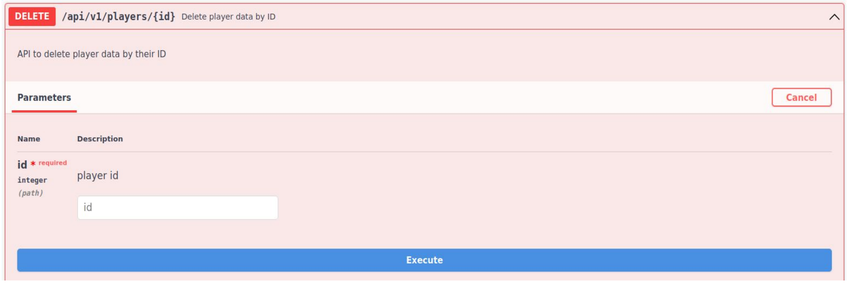


Responses

Code	Description	Links
200	Show response to indicate that the operation has succeed Media type: application/json Controls Accept header. Example Value | Schema `{ "result": "Success", "message": "[Example] Player data is successfully updated" }`	*No links*
400	Show response if there's error in user side. Please recheck the input of request bodies or parameters. Media type: application/json Example Value | Schema `{ "result": "Failed", "message": "[Example] Failed to upload" }`	*No links*
500	Show response if there's error in server side. Media type: application/json Example Value | Schema `{ "result": "Failed", "message": "[Example] Failed to upload" }`	*No links*

API to update player data by id
Endpoint: /api/v1/players/:id
Method: PUT
ID, username, email, and password (min. 6 character) are required.

API DOCUMENTATION

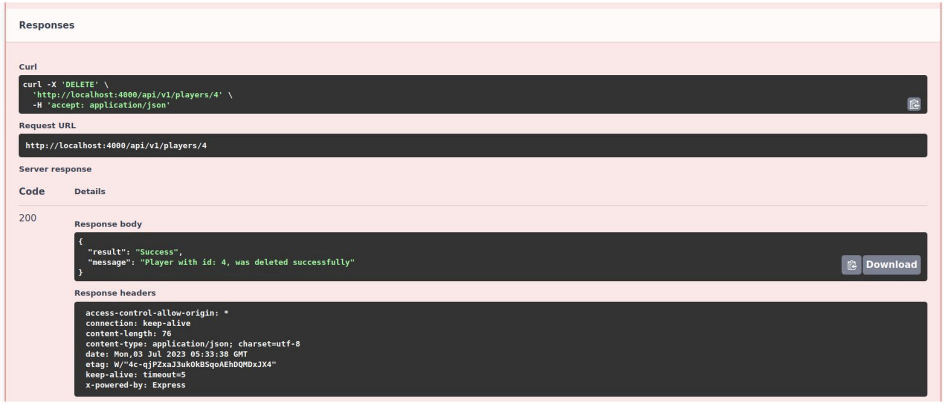


DELETE /api/v1/players/{id} Delete player data by ID

API to delete player data by their ID

Parameters

Name	Description
id * required integer (path)	player id



Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:4000/api/v1/players/4' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:4000/api/v1/players/4
```

Server response

Code Details

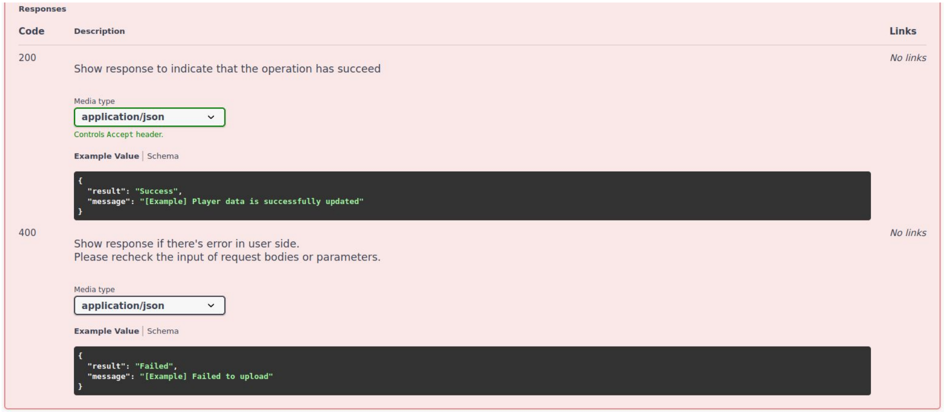
200

Response body

```
{
  "result": "Success",
  "message": "Player with id: 4, was deleted successfully"
}
```

Response headers

```
 access-control-allow-origin: *
 connection: keep-alive
 content-length: 76
 content-type: application/json; charset=utf-8
 date: Mon,03 Jul 2023 05:33:38 GMT
 etag: W/"4c-qjPZxaJ3ukOkBSqoAEhDQMDxJX4"
 keep-alive: timeout=5
 x-powered-by: Express
```



Responses

Code	Description	Links
200	Show response to indicate that the operation has succeed	No links
400	Show response if there's error in user side. Please recheck the input of request bodies or parameters.	No links

200 — Media type: application/json — Controls Accept header.

Example Value | Schema

```
{
  "result": "Success",
  "message": "[Example] Player data is successfully updated"
}
```

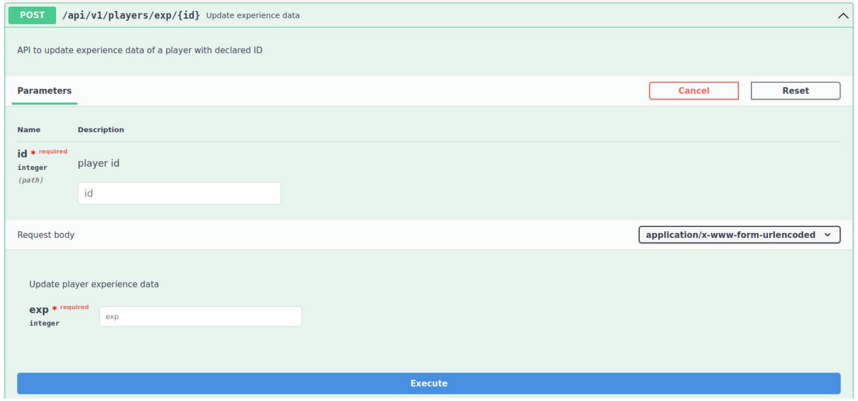
400 — Media type: application/json

Example Value | Schema

```
{
  "result": "Failed",
  "message": "[Example] Failed to upload"
}
```

API to delete player data by id
Endpoint: /api/v1/players/:id
Method: DELETE
ID is required.

API DOCUMENTATION



POST `/api/v1/players/exp/{id}` Update experience data

API to update experience data of a player with declared ID

Parameters — Cancel | Reset

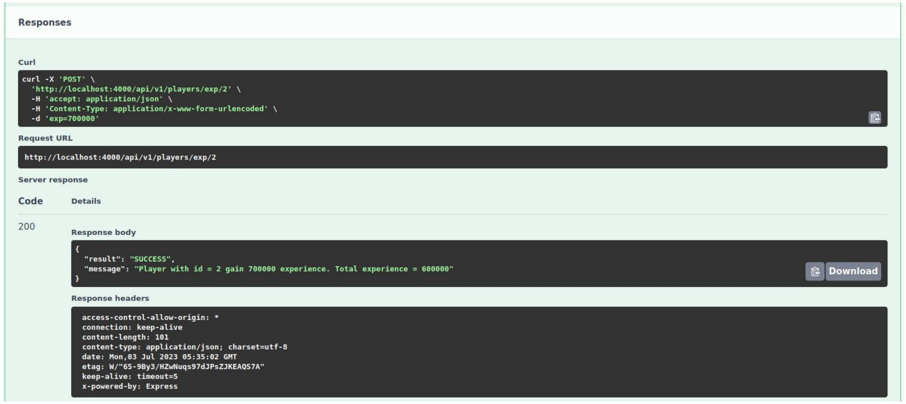
Name	Description
id * required integer *(path)*	player id id

Request body — application/x-www-form-urlencoded

Update player experience data

exp * required
integer — exp

Execute



Responses

Curl

```
curl -X 'POST' \
  'http://localhost:4000/api/v1/players/exp/2' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d 'exp=700000'
```

Request URL

```
http://localhost:4000/api/v1/players/exp/2
```

Server response

Code — **Details**

200

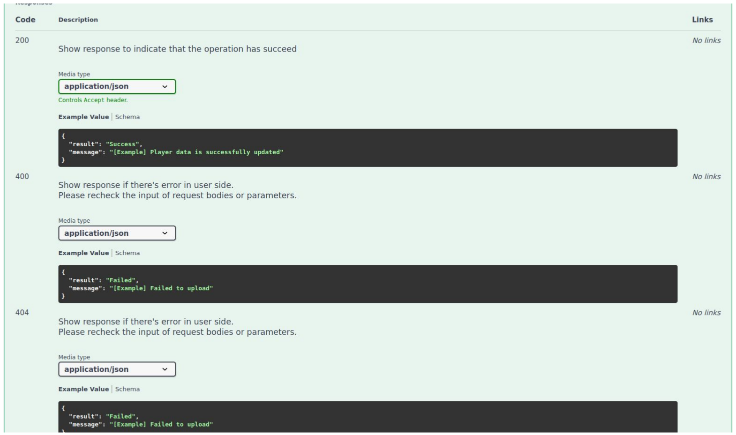
Response body

```
{
  "result": "SUCCESS",
  "message": "Player with id = 2 gain 700000 experience. Total experience = 600000"
}
```

Response headers

```
 access-control-allow-origin: *
 connection: keep-alive
 content-length: 101
 content-type: application/json; charset=utf-8
 date: Mon,03 Jul 2023 05:35:02 GMT
 etag: W/"65-9By3/HZwNuqs97dJPsZJKEAQS7A"
 keep-alive: timeout=5
 x-powered-by: Express
```

API to update experience of player by id
Endpoint: /api/v1/players/exp/:id
Method: POST
ID and (new)Experience are required.



Code	Description	Links
200	Show response to indicate that the operation has succeed Media type: application/json Controls Accept header. **Example Value** | Schema `{ "result": "Success", "message": "[Example] Player data is successfully updated" }`	*No links*
400	Show response if there's error in user side. Please recheck the input of request bodies or parameters. Media type: application/json **Example Value** | Schema `{ "result": "Failed", "message": "[Example] Failed to upload" }`	*No links*
404	Show response if there's error in user side. Please recheck the input of request bodies or parameters. Media type: application/json **Example Value** | Schema `{ "result": "Failed", "message": "[Example] Failed to upload"`	*No links*

TASK CHECKLIST

ACTION	CHECKLIST
Clone project from github →work on server folder	Done
Initialize server connection using Sequelize	Done
Create documentation including: HTTP Method, URL, URL Parameter, Query Parameter, Request Body, format response	Done
Put all documentation in one routing	Done

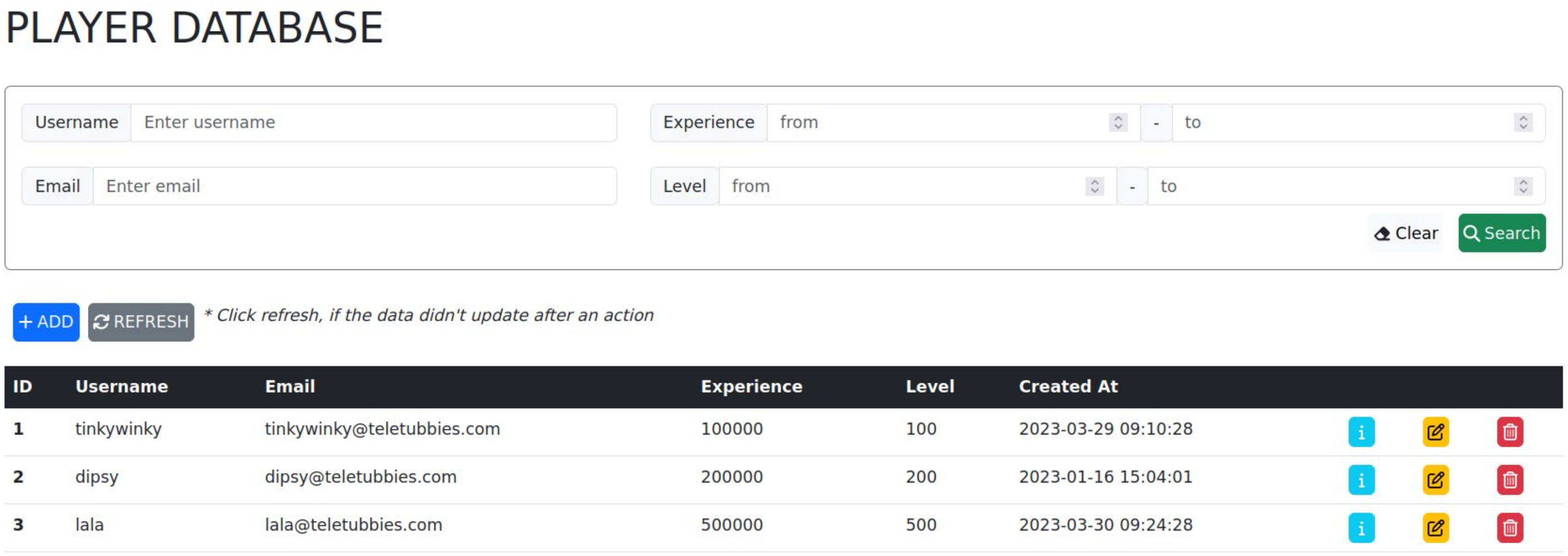
REACT CLIENT

PREPARATION

- Project initialization using boilerplate: $ npx create-react-app client
- For CRUD process, we will use dummy data, stored in */lib/dummyData.json.*
- React client runs in url **http://localhost:3000/**

REACT CLIENT

Dashboard UI



PLAYER DATABASE

Username	Enter username		Experience	from	-	to
Email	Enter email		Level	from	-	to

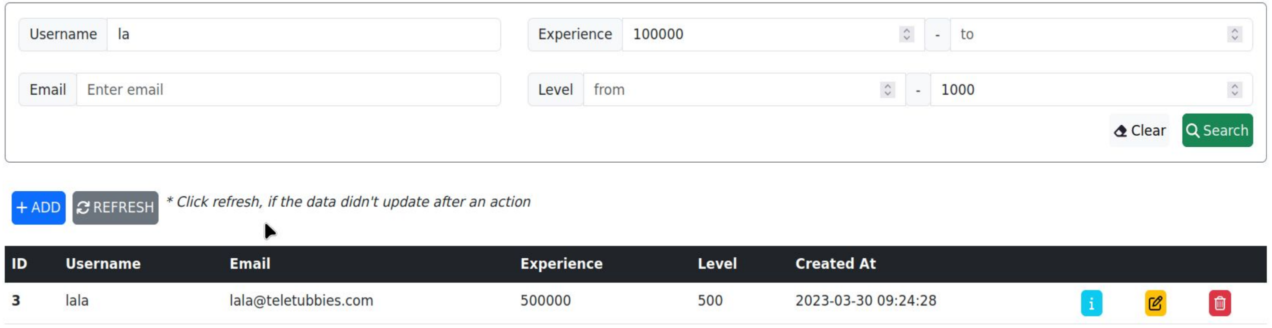
** Click refresh, if the data didn't update after an action*

ID	Username	Email	Experience	Level	Created At
1	tinkywinky	tinkywinky@teletubbies.com	100000	100	2023-03-29 09:10:28
2	dipsy	dipsy@teletubbies.com	200000	200	2023-01-16 15:04:01
3	lala	lala@teletubbies.com	500000	500	2023-03-30 09:24:28

REACT CLIENT

SEARCH

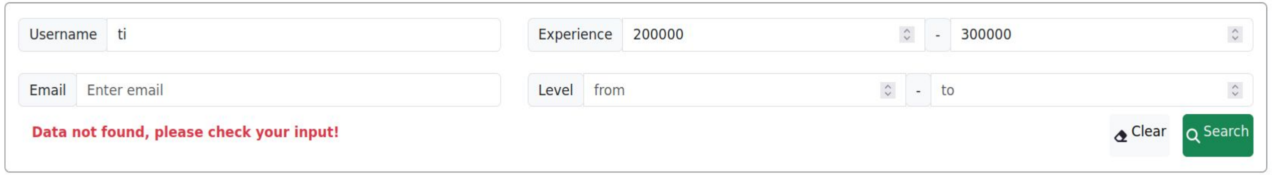
- Without any input, clicking **Search** button will show all data
- String input can accommodate substring search. Numeric range can accept only one input (*from* only or *to* only)
- If the parameter return no result, it will show an alert text
- **Clear** button is available to clear the input fields without changing the search result.



Username	la	Experience	100000	-	to
Email	Enter email	Level	from	-	1000

* Click refresh, if the data didn't update after an action

ID	Username	Email	Experience	Level	Created At
3	lala	lala@teletubbies.com	500000	500	2023-03-30 09:24:28



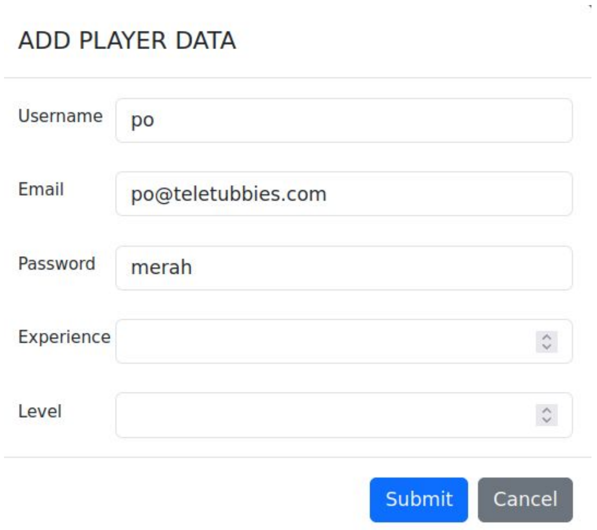
Username	ti	Experience	200000	-	300000
Email	Enter email	Level	from	-	to

Data not found, please check your input!

REACT CLIENT

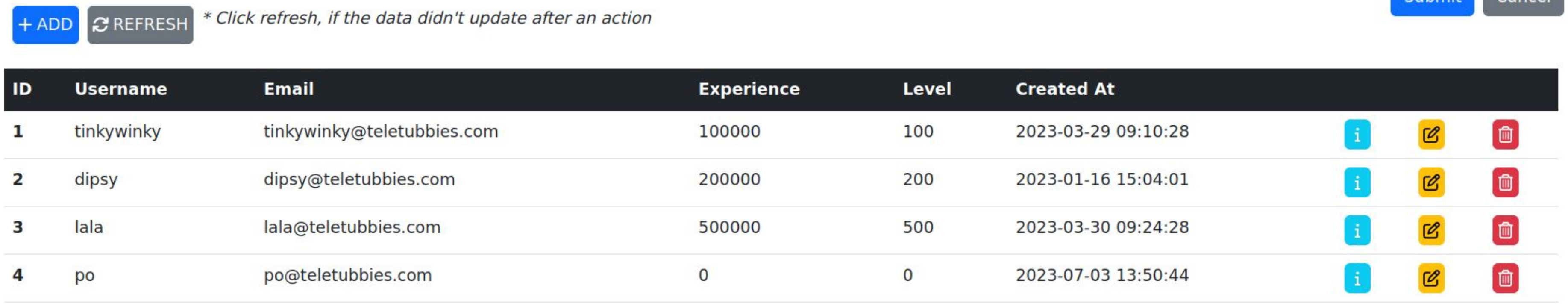
ADD

- **Add** button will open a form modal to submit the data.
- For this Client side, there is no validation or password encryption action.
- If the data is not shown on main table after **Submit**, click **Refresh** button.



ADD PLAYER DATA

Username	po
Email	po@teletubbies.com
Password	merah
Experience	
Level	



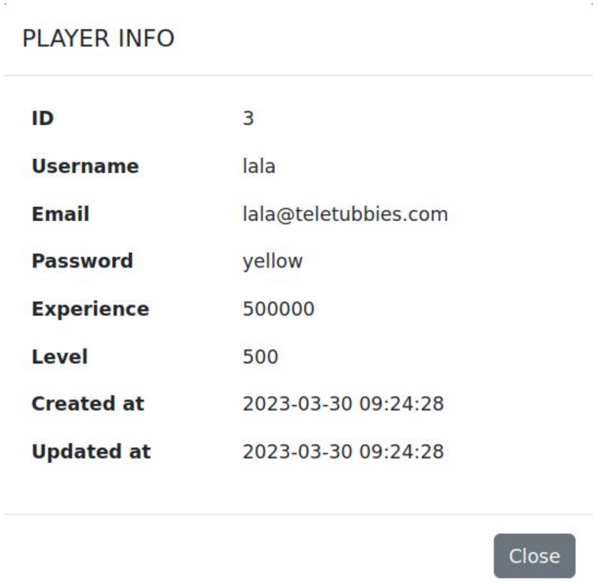
** Click refresh, if the data didn't update after an action*

ID	Username	Email	Experience	Level	Created At
1	tinkywinky	tinkywinky@teletubbies.com	100000	100	2023-03-29 09:10:28
2	dipsy	dipsy@teletubbies.com	200000	200	2023-01-16 15:04:01
3	lala	lala@teletubbies.com	500000	500	2023-03-30 09:24:28
4	po	po@teletubbies.com	0	0	2023-07-03 13:50:44

REACT CLIENT

INFO

- **Info** button will show all data a player has.



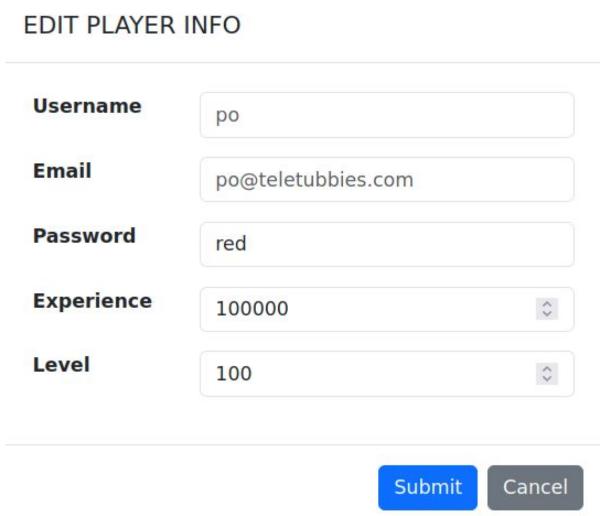
PLAYER INFO

ID	3
Username	lala
Email	lala@teletubbies.com
Password	yellow
Experience	500000
Level	500
Created at	2023-03-30 09:24:28
Updated at	2023-03-30 09:24:28

REACT CLIENT

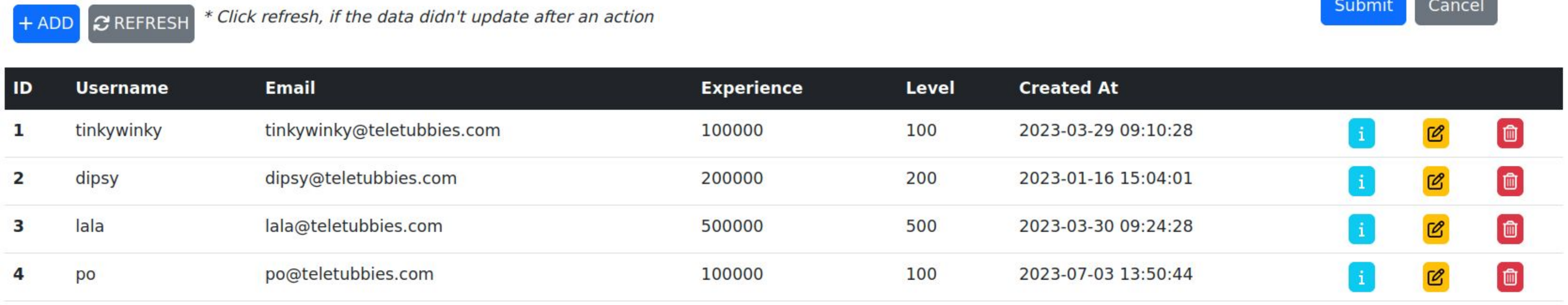
EDIT

- **Edit** button will show form to edit player data.



EDIT PLAYER INFO

Username	po
Email	po@teletubbies.com
Password	red
Experience	100000
Level	100



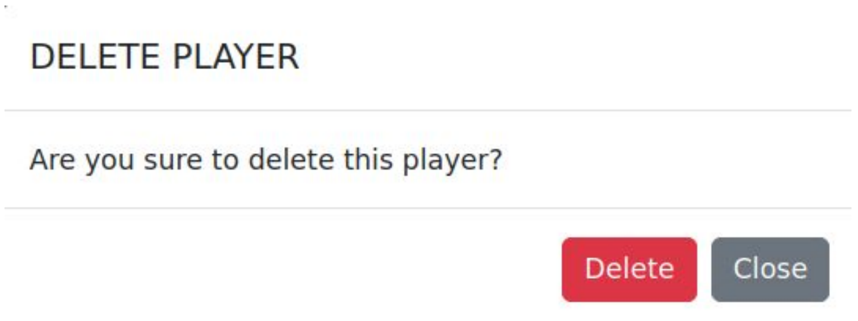
** Click refresh, if the data didn't update after an action*

ID	Username	Email	Experience	Level	Created At
1	tinkywinky	tinkywinky@teletubbies.com	100000	100	2023-03-29 09:10:28
2	dipsy	dipsy@teletubbies.com	200000	200	2023-01-16 15:04:01
3	lala	lala@teletubbies.com	500000	500	2023-03-30 09:24:28
4	po	po@teletubbies.com	100000	100	2023-07-03 13:50:44

REACT CLIENT

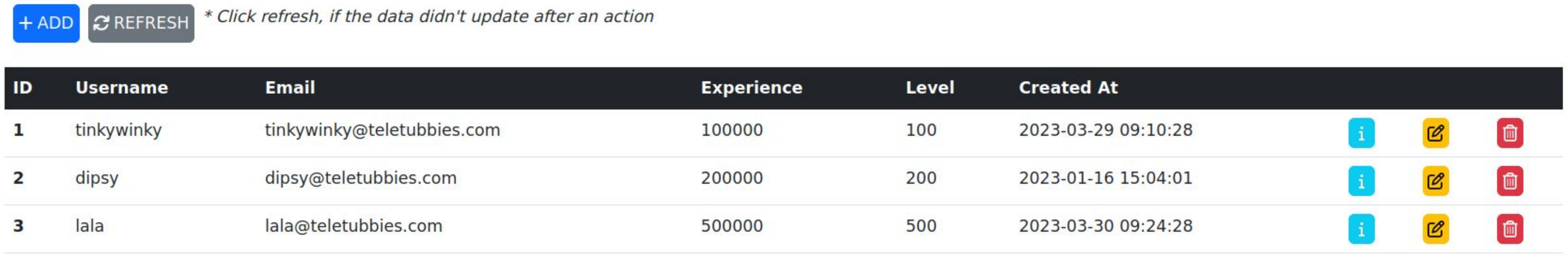
DELETE

- **Delete** button will delete the selected player data



DELETE PLAYER

Are you sure to delete this player?



** Click refresh, if the data didn't update after an action*

ID	Username	Email	Experience	Level	Created At
1	tinkywinky	tinkywinky@teletubbies.com	100000	100	2023-03-29 09:10:28
2	dipsy	dipsy@teletubbies.com	200000	200	2023-01-16 15:04:01
3	lala	lala@teletubbies.com	500000	500	2023-03-30 09:24:28

TASK CHECKLIST

ACTION	CHECKLIST
Initialize React Client	Done
Create UI for Add, Edit, and Search player data based on username, email, experience and level.	Done
Show the result of all CRUD process	Done