第一届山东省职业技能大赛
网站技术项目样题

第一届山东省职业技能大赛组委会办公室技术工作组

2023 年 5 月

TEST PROJECT

Server Side

Submitted by:

Assessment Browser:
Google Chrome

Competition Time:
Part I - 3 hours
Part II - 3 hours

INTRODUCTION

With the popularization of the Internet, informatization construction has become more and more popular in all aspects of life. Carrying out informatization construction in an enterprise can assist enterprise employees in efficient collaboration and improve enterprise production efficiency. A company called "Z Camp" has always been committed to serving primary and secondary schools to carry out student quality development activities. With the growing scale of the company, the use of email communication has become very inefficient.

The module is divided into two parts. In the first part, you need to complete the back-end API development work according to the needs, and in the second part, use the API developed in the first part to make front-end application functions.

DESCRIPTION OF PROJECT AND TASKS

First 3 hours – Api Development

The description for the first phase of the project is listed below. The first task is to create a restful web service API that can be used by the front end to communicate the data. In the absence of special instructions, Status of 422 is used to process the case where the required field for the request is empty.

Staff Panel

1. Auth

a. Login (/login)
Description: Admin and employee can login.
Request Method: POST
Request Parameters:
- email
- password

Response Result:
- Data for success:
  - Status Code: 200
  - Body:
    - msg: “success”
    - data:
      - id
      - email
      - token
      - scope (“course” , “training”)
      - is_admin (1 , 0)
- Data for wrong data:
  - Status Code: 422
  - Body:
    - msg: “data cannot be processed”
- Data for wrong user credential:
  - Status Code: 401
  - Body:
    - msg: “user credentials are invalid”

Body Result Sample for Success:

```
{
    “msg”: “success” ,
    “data”: {
        “id”: 1 ,
        “email”: “sample@email.com” ,
        “token”: “AUTHORIZATION_TOKEN” ,
        “scope”: “course”,
        “is_admin”: 1
    }
}
```

b. Logout (/logout?token={token})
Description: Admin and employee can logout.
Request Method: POST
Response Result:

- Data for success:
  - Status Code: 200
  - Body:
    - msg: “success”

- Data for unauthorized:
  - Status Code: 401
  - Body:
    - msg: “unauthorized”

2. Dashboard
   a. Get Datetime (/datetime?token={token})
   Description: Admin and employee can get current server datetime.
   Request Method: GET
   Response Result:
   - Data for success:
     - Status Code: 200
     - Body:
       - msg: ”success”
       - data:
         - datetime (format：[YYYY-mm-dd HH:ii:ss])

   - Data for Unauthorized:
     - Status Code: 401
     - Body:
       - msg: “unauthorized”

   Body Result Sample for Success:

```
{
    “msg”: “success” ,
    “data”: {
        “datetime”: “2020-12-10 10:00:00”
    }
}
```

3. Event
    a. Get All Events (/event?token={token})
    Description: Admin can gets all events
    Request Method: GET
    Response Result:
    - Data for success :
        - Status Code: 200
        - Body:
            - msg: ”success”
            - data(array):
                - id
                - name
                - code
                - type (“course” / “training”)
                - address
                - date (format：[YYYY-mm-dd])
                - permission (“public” / “private”)
                - status (“pending” / “accepted” / “finished”)

    - Data for Unauthorized:
        - Status Code: 401
        - Body:
            - msg: “unauthorized”

    Body Result Sample for Success:

```
{
    “msg”: “success” ,
    “data”: [
        {
            “id”: 1,
            “name”: “event name”,
            “code”: “EVENT_CODE”,
            “type”: “course”,
            “address”: “event address”,
            “date”: “2020-12-11”,
            “permission”: “public”,
            “status”: “pending”,
        },
        {
            “id”: 2,
            “name”: “event name”,
            “code”: “EVENT_CODE”,
            “type”: “course”,
            “address”: “event address”,
            “date”: “2020-12-11”,
            “permission”: “public”,
            “status”: “pending”,
        },
        ...
    ]
```

}

b. Get My Events (/event/my?token={token})

Description: Current login employee (or admin) can get all events that assigned to him.

Request Method: GET

Response Result:

- Data for success:
  - Status Code: 200
  - Body:
    - msg: “success”
    - data(array):
      - id
      - name
      - code
      - type (“course” / “training”)
      - address
      - date (format：[YYYY-mm-dd])
      - permission (“public” / “private”)
      - status (“pending” / “accepted” / “finished”)
      - response (“pending” / “accepted”) (employee’s response to this event)
- Data for unauthorized:
  - Status Code: 401
  - Body:
    - msg: “unauthorized”

Body Result Sample for Success:

```
{
    “msg”: “success” ,
    “data”: [
        {
            “id”: 1,
            “name”: “event name”,
            “code”: “EVENT_CODE”,
            “type”: “course”,
            “address”: “event address”,
            “date”: “2020-12-11”,
            “permission”: “public”,
            “status”: “pending”,
            “response”: “accepted”,
        },
        {
            “id”: 2,
            “name”: “event name”,
            “code”: “EVENT_CODE”,
            “type”: “course”,
            “address”: “event address”,
            “date”: “2020-12-11”,
            “permission”: “public”,
            “status”: “pending”,
            “response”: “pending”,
        },
        ...
    ]
```

}

c. Get Event by ID (/event/{event_id}?token={token})
Description: Admin and employee can get event info by using ID, and return 404 if can't find the event.
Request Method: GET
Response Result:

- Data for success:
  - Status Code: 200
  - Body:
    - msg: "success"
    - data:
      - id
      - name
      - code
      - type ("course" / "training")
      - photo
      - address
      - date(format：[YYYY-mm-dd])
      - permission ("public" / "private")
      - status ("pending" / "accepted" / "finished")
      - employees(array):
        - id
        - email
        - photo
        - scope ("course" / "training")
        - firstname
        - lastname
        - response ("pending" / "accepted") (employee's response to this event)
      - client:
        - id
        - corporate_name
        - firstname
        - lastname
        - phone
        - email

- Data for unauthorized:
  - Status Code: 401
  - Body:
    - msg: "unauthorized"

- Data for event not found:
  - Status Code: 404
  - Body:
    - msg: "not found"

Body Result Sample for Success:

```
{
    "msg": "success" ,
```

```
        “data”: {
                “id”: 1,
                “name”: “event name”,
                “code”: “EVENT_CODE”,
                “type”: “course”,
                “photo”: “IMAGE_URI”,
                “address”: “event address”,
                “date”: “2020-12-11”,
                “permission”: “public”,
                “status”: “pending”,
                “employees”: [
                        {
                                “id”: 1,
                                “email”: “sample1@email.com” ,
                                “photo”: “IMAGE_URI”,
                                “scope”: “course”,
                                “firstname”: “firstname”,
                                “lastname”: “lastname”,
                                “response”: “pending”
                        },
                        {
                                “id”: 2,
                                “email”: “sample2@email.com” ,
                                “photo”: “IMAGE_URI”,
                                “scope”: “training”,
                                “firstname”: “firstname”,
                                “lastname”: “lastname”,
                                “response”: “accepted”
                        },
                        ...
                ],
                “client”: {
                        “id”: 1,
                        “corporate_name”: “corporate”,
                        “firstname”: “firstname”,
                        “lastname”: “lastname”,
                        “phone”: “123456789”,
                        “email”: “cor_sample@email.com”
                }
        },
}
```


d. Add New Event (/event?token={token})
Description: Admin can add an event. The date in request parameter must be tomorrow or later, Employee id in the employee_ids must from an existing employee, client_id must from an existing client, the scope of the assigned employee must match the event’s type, otherwise 422 will be returned.
Request Method: POST
Request Parameters:
- name
- type (“course” / “training”)
- photo

- address
- date(format：[YYYY-mm-dd])
- client_id
- employee_ids (array to string,value is similar with: “[1,2,3,4]”)
- permission (“public” / “private”)

Response Result:

- Data for success:
  - Status Code: 200
  - Body:
    - msg: “success”
    - data:
      - id
- Data for unauthorized:
  - Status Code: 401
  - Body:
    - msg: “unauthorized”
- Data for wrong data:
  - Status Code: 422
  - Body:
    - msg: “data cannot be processed”

e. Update Employees in Event (/event/employees/{event_id}?token={token})
Description: Admin can update an event’s employees. Employee id in the employee_ids must from an existing employee,the scope of the assigned employee must match the event’s type, otherwise 422 will be returned.

Request Method: PATCH
Request Parameters:

- employee_ids (a nullable array to string,value is similar with: “[1,2,3,4]”)

Response Result:

- Data for success:
  - Status Code: 200
  - Body:
    - msg: “success”
- Data for unauthorized:
  - Status Code: 401
  - Body:
    - msg: “unauthorized”
- Data for event not found:
  - Status Code: 404
  - Body:
    - msg: “not found”
- Data for wrong data:
  - Status Code: 422
  - Body:
    - msg: “data cannot be processed”

f. Update Client in Event (/event/client/{event_id}?token={token})

Description: Admin can update an event's client. The client_id must from an existing client, otherwise 422 will be returned.

Request Method: PATCH
Request Parameters:
- client_id

Response Result:
- Data for success:
  - Status Code: 200
  - Body:
    - msg: "success"
- Data for unauthorized:
  - Status Code: 401
  - Body:
    - msg: "unauthorized"
- Data for event not found:
  - Status Code: 404
  - Body:
    - msg: "not found"
- Data for wrong data:
  - Status Code: 422
  - Body:
    - msg: "data cannot be processed"

g. Delete Event (/event/{event_id}?token={token})
Description: Admin can delete an event. All related client and employees' relationships will be removed. All enrollments under this event will be removed as well.
Request Method: DELETE
Response Result:
- Data for success:
  - Status Code: 200
  - Body:
    - msg: "success"

- Data for Unauthorized:
  - Status Code: 401
  - Body:
    - msg: "unauthorized"

- Data for Event Not Found:
  - Status Code: 404
  - Body:
    - msg: "not found"

h. Get All Events that has un-replied messages (/event/unreplied?token={token})
Description: Current login employee (or admin) can get all the events that has un-replied messages.
Request Method: GET
Response Result:
- Data for success:

- Status Code: 200
- Body:
  - msg: “success”
  - data(array):
    - id (id of the event)
    - name (name of the event)
    - corporate_name (corporate`s name of the client)
    - last_message(last message of the un-replied messages):
      - id
      - content
      - is_reply (the return value must be 0)
    - message_count (the number of un-replied messages)

- Data for Unauthorized:
  - Status Code: 401
  - Body:
    - msg: “unauthorized”

Body Result Sample for Success:

```
{
    “msg”: “success” ,
    “data”: [
        {
            “id”: 1,
            “name”: “event name”,
            “corporate_name”: “corporate name”,
            “last_message”: {
                “id”: 1,
                “content”: “Lorem ipsum dolor sit.”,
                “is_reply”: 0,
            }
            “message_count”: 2,
        },
        {
            “id”: 2,
            “name”: “event name”,
            “corporate_name”: “corporate name”,
            “last_message”: {
                “id”: 2,
                “content”: “Lorem ipsum dolor sit.”,
                “is_reply”: 0,
            }
            “message_count”: 2,
        },
        ...
    ]
}
```

4. Employee
   a. Get All Employees (/employee?token={token})
   Description: Admin can get all employees.
   Request Method: GET
   Response Result:
   - Data for success:

- Status Code: 200
- Body:
    - msg: “success”
    - data(array):
        - id
        - email
        - photo
        - scope (“course” or “training”)
        - firstname
        - lastname

- Data for Unauthorized:
    - Status Code: 401
    - Body:
        - msg: “unauthorized”

Body Result Sample for Success:

```
{
    “msg”: “success” ,
    “data”: [
        {
            “id”: 1,
            “email”: “sample1@email.com” ,
            “photo”: “IMAGE_URI”,
            “scope”: “course”,
            “firstname”: “firstname”,
            “lastname”: “lastname”
        },
        {
            “id”: 2,
            “email”: “sample2@email.com” ,
            “photo”: “IMAGE_URI”,
            “scope”: “training”,
            “firstname”: “firstname”,
            “lastname”: “lastname”
        },
        ...
    ],
}
```

b. Get Employee by ID (/employee/{employee_id}?token={token})
Description: Admin can get an employee’s info by employee_id.
Request Method: GET
Response Result:

- Data for success:
    - Status Code: 200
    - Body:
        - msg: “success”
        - data
            - id
            - email

- photo
- scope (“course” or “training”)
- firstname
- lastname

- Data for Unauthorized:
  - Status Code: 401
  - Body:
    - msg: “unauthorized”

- Data for Not Found:
  - Status Code: 404
  - Body:
    - msg: “not found”

Body Result Sample for Success:

```
{
    “msg”: “success” ,
    “data”: {
        “id”: 1,
        “email”: “sample1@email.com” ,
        “photo”: “IMAGE_URI”,
        “scope”: “course”,
        “firstname”: “firstname”,
        “lastname”: “lastname”,
    },
}
```

c. Add New Employee (/employee?token={token})
Description: Admin can create a new employee (or admin)
Request Method: POST
Request Parameter:
- email
- photo
- scope (“course” , “training”)
- password
- firstname
- lastname
- is_admin (1,0)

Response Result:
- Data for success:
  - Status Code: 200
  - Body:
    - msg: “success”
    - data
      - id

- Data for unauthorized:
  - Status Code: 401
  - Body:
    - msg: “unauthorized”

- Data for wrong data:
    - Status Code: 422
    - Body:
        - msg: “data cannot be processed”

- Data for repeat email:
    - Status Code: 422
    - Body:
        - msg: “This email address has already been taken”

d. Delete Employee (/employee/{employee_id}?token={token})
Description: Admin can delete an employee (or admin) except for himself（401）. Relationships between this employee and other events will be removed. (Events won’t be removed)
Request Method: DELETE
Response Result:
- Data for success:
    - Status Code: 200
    - Body:
        - msg: “success”

- Data for Unauthorized:
    - Status Code: 401
    - Body:
        - msg: “unauthorized”

- Data for Not Found:
    - Status Code: 404
    - Body:
        - msg: “not found”

e. Employee Accept an Event(/employee/accept/{event_id}?token={token})
Description: Admin and employee can accept an event. If the employee(or admin) is not assigned to the event, return 401;If the event status is not pending, return 422;
Request Method: POST
Response Result:
- Data for success
    - Status Code: 200
    - Body:
        - msg: “success”
        - data:
            - id (event id)
            - status (If every employees in this event accepted this event, the returned status have to be “accepted”)

- Data for unauthorized:
    - Status Code: 401
    - Body:
        - msg: “unauthorized”
- Data for not assigned:
    - Status Code: 401
    - Body:

- msg: “the employee is not assigned to the event”

- Data for not found:
  - Status Code: 404
  - Body:
    - msg: “not found”
- Data for wrong status:
  - Status Code: 422
  - Body:
    - msg: “This event status is not pending”

f. Employee Finish an Event (/employee/finish/{event_id}?token={token})
Description: Admin and employee can finish an event. If the employee is not assigned to the event, return 401; If the event’s status is not “accepted”, return 422; Admin can finish all events;

Request Method: POST
Response Result:

- Data for success
  - Status Code: 200
  - Body:
    - msg: “success”
    - data:
      - id (event id)
      - status (the return value must be “finished”)
- Data for unauthorized:
  - Status Code: 401
  - Body:
    - msg: “unauthorized”
- Data for not assigned:
  - Status Code: 401
  - Body:
    - msg: “the employee is not assigned to the event”
- Data for event not found (or no relation between event and employee):
  - Status Code: 404
  - Body:
    - msg: “not found”
- Data for wrong data:
  - Status Code: 422
  - Body:
    - msg: “the event’s status is not ‘accepted’”

5. Client
   a. Get All Clients (/client?token={token})
   Description: Admin can get all clients.
   Request Method: GET
   Response Result:
   - Data for success:
     - Status Code: 200
     - Body:
       - msg: “success”
       - data(array):
         - id
         - corporate_name
         - firstname
         - lastname

- phone
- email

- Data for Unauthorized:
  - Status Code: 401
  - Body:
    - msg: “unauthorized”

Body Result Sample for Success:

```
{
    “msg”: “success” ,
    “data”: [
        {
            “id”: 1,
            “corporate_name”: “corporate name”,
            “firstname”: “firstname”,
            “lastname”: “lastname”,
            “phone”: “123456789”,
            “email”: “cor_sample@email.com”
        },
        {
            “id”: 2,
            “corporate_name”: “corporate name”,
            “firstname”: “firstname”,
            “lastname”: “lastname”,
            “phone”: “123456789”,
            “email”: “cor_sample@email.com”
        }
        ...
    ],
}
```

b. Get Client by ID (/client/{client_id}?token={token})
Description: Admin can get an client’s info by client_id.
Request Method: GET
Response Result:

- Data for success:
  - Status Code: 200
  - Body:
    - msg: “success”
    - data
      - id
      - corporate_name
      - firstname
      - lastname
      - phone
      - email

- Data for Unauthorized:
  - Status Code: 401
  - Body:
    - msg: “unauthorized"

- Data for not found:
    - Status Code: 404
    - Body:
        - msg: “not found”

Body Result Sample for Success:

```
{
    “msg”: “success” ,
    “data”: {
        “id”: 1,
        “corporate_name”: “corporate name”,
        “firstname”: “firstname”,
        “lastname”: “lastname”,
        “phone”: “123456789”,
        “email”: “cor_sample@email.com”
    },
}
```

c. Add New Client (/client?token={token})
Description: Admin can add a new client. If the corporate_name is repeated, a 422 error will be returned.
Request Method: POST`
Request Parameter:
- corporate_name
- firstname
- lastname
- phone
- email

Response Result:
- Data for success:
    - Status Code: 200
    - Body:
        - msg: “success”
        - data
            - id

- Data for unauthorized:
    - Status Code: 401
    - Body:
        - msg: “unauthorized”

- Data for wrong data:
    - Status Code: 422
    - Body:
        - msg: “data cannot be processed”

- Data for repeat corporate name:
    - Status Code: 422
    - Body:
        - msg: “This Corporate name has already been taken”

d. Delete Client (/client/{client_id}?token={token})
Description: Admin can delete a client. If this client has related event, return 422.
Request Method: DELETE
Response Result:

- Data for success:
  - Status Code: 200
  - Body:
    - msg: “success”

- Data for Unauthorized:
  - Status Code: 401
  - Body:
    - msg: “unauthorized”

- Data for Not Found:
  - Status Code: 404
  - Body:
    - msg: “not found”

- Data for an occupied client:
  - Status Code: 422
  - Body:
    - msg: “Client still has related events”

User Panel

1. Event
   a. Get All Event – frontend (/event-list)
   Description: Get all events in the user panel.
   Request Method: GET
   Response Result:
   - Data for success:
     - Status Code: 200
     - Body:
       - msg: “success”
       - data(array):
         - id
         - name
         - type (“course” or “training”)
         - photo
         - date (format:[YYYY-mm-dd])
         - permission (“public” or “private”)

   Body Result Sample for Success:

```
{
        “msg”: “success” ,
        “data”: [
                {
```

```
                    “id”: 1,
                    “name”: “event name”,
                    “type”: “course”,
                    “photo”: “IMAGE_URI”,
                    “date”: “2020-12-11”,
                    “permission”: “public”,
                },
                {
                    “id”: 2,
                    “name”: “event name”,
                    “type”: “training”,
                    “photo”: “IMAGE_URI”,
                    “date”: “2020-12-11”,
                    “permission”: “public”,
                },
                ...
            ]

    }
```

b. Get Event Preview by Code (/event/preview/{event_code})
Description: User can get an event’s preview by event_code(case insensitive).
Request Method: GET
Response Result:

- Data for success:
  - Status Code: 200
  - Body:
    - msg: “success”,
    - data
      - id
      - name
      - type (“course” or “training”)
      - photo
      - address
      - corporate_name
      - date (format:[YYYY-mm-dd])

- Data for event not found:
  - Status Code: 404
  - Body:
    - msg: “not found”

Body Result Sample for Success:

```
{
    “msg”: “success” ,
    “data”: {
        “id”: 1,
        “name”: “event name”,
        “type”: “course”,
        “photo”: “IMAGE_URI”,
        “address”: “event address”,
        “corporate_name”: “corporate name”,
        “date”: “2020-12-11”,
    }
```

}

c. Get Event Detail by Code & Phone (/event/detail/{event_code}/{client_phone})
Description: User can get an event's detail info by using event_code(case insensitive) and client_phone. If the event's client_phone doesn't match the passed client_phone, return 422.
Request Method: GET
Response Result:

- Data for success:
  - Status Code: 200
  - Body:
    - msg: "success",
    - data
      - id
      - name
      - photo
      - address
      - date (format:[YYYY-mm-dd])
      - employees(array):
        - id
        - email
        - photo
        - scope ("course" or "training")
        - firstname
        - lastname
        - response(pending,accepted)
      - client:
        - id
        - corporate_name
        - firstname
        - lastname
        - phone
        - email
      - permission ("public" or "private")
      - type ("course" or "training")
      - status ("pending", "accepted" , "finished")

- Data for event not found:
  - Status Code: 404
  - Body:
    - msg: "not found"

- Data for event phone not match:
  - Status Code: 422
  - Body:
    - msg: "the phone number does not match the event code"

Body Result Sample for Success:

```
{
    "msg": "success" ,
    "data": {
```

```
                    “id”: 1,
                    “name”: “event name”,
                    “photo”: “IMAGE_URI”,
                    “address”: “event address”,
                    “date”: “2020-12-11”,
                    “employees”: [
                            {
                                    “id”: 1,
                                    “email”: “sample1@email.com” ,
                                    “photo”: “IMAGE_URI”,
                                    “scope”: “course”,
                                    “firstname”: “firstname”,
                                    “lastname”: “lastname”
                                    “response”: “pending”

                            },
                            {
                                    “id”: 2,
                                    “email”: “sample2@email.com” ,
                                    “photo”: “IMAGE_URI”,
                                    “scope”: “training”,
                                    “firstname”: “firstname”,
                                    “lastname”: “lastname”
                                    “response”: “accepted”

                            },
                            ...
                    ],
                    “client”: {
                            “id”: 1,
                            “corporate_name”: “corporate”,
                            “firstname”: “firstname”,
                            “lastname”: “lastname”,
                            “phone”: “123456789”,
                            “email”: “cor_sample@email.com”
                    }
                    “permission”: “public”,
                    “type”: “course”,
                    “status”: “pending”,
            },
    }
```


d. Check Event(/event/check/{event_id}/{event_code})
Description: Check if the code (case insensitive) matches the given event.
Request Method: POST
- Request Parameter:

Response Result:
- Data for success:
  - Status Code: 200
  - Body:
    - msg: “success”
- Data for event not found:
  - Status Code: 404

        - Body:
            - msg: “not found”
    - Data for wrong event code:
        - Status Code: 401
        - Body:
            - msg: “the event code is not correct”

2. Enrollments
    a. Enroll Event (/enroll/{event_id})
    Description: User can enroll an event. If the same event have a repeat email address, 422 will be returned.
    Request Method: POST
    Request Parameter:
    - email
    - phone
    - firstname
    - lastname

    Response Result:
    - Data for success:
        - Status Code: 200
        - Body
            - msg: “success”
            - data
                - id
    - Data for wrong data:
        - Status Code: 422
        - Body:
            - msg: “data cannot be processed”
    - Data for event not found:
        - Status Code: 404
        - Body:
            - msg: “not found”
    - Data for repeat email (Cannot enroll a same email address for same event):
        - Status Code: 422
        - Body:
            - msg: “This email address has already been taken”

    b. Get All enrollments by Email and Phone (/enroll/{email}/{phone})
    Description: User can get history enrollments by using email and phone.
    Request Method: GET
    Response Result:
    - Data for success:
        - Status Code: 200
        - Body
            - msg: “success”
            - data (array)
                - id(enrollment id)
                - firstname(enrollment firstname)
                - lastname(enrollment lastname)
                - event:

- name
- photo
- date

- Data for email not found:
  - Status Code: 404
  - Body:
    - msg: “not found”

Body Result Sample for Success:

```
{
    “msg”: “success” ,
    “data”:  [
        {
            “id”: 1,
            “firstname”: “firstname”,
            “lastname”: “lastname”,
            “event”: {
                “name”: “event name”,
                “photo”:“IMAGE_URI”,
                 “date”: “2020-12-10”,
            },
        },
        {
            “id”: 2,
            “firstname”: “firstname”,
            “lastname”: “lastname”,
            “event”: {
                “name”: “event name”,
                “photo”:“IMAGE_URI”,
                 “date”: “2020-12-10”,
            },
        },
        ...
    ],
}
```

3. Message
   a. Get Messages (/message/{event_id})
   Description: Get All Messages Under Specific Event. If ‘is_reply’ equals 1, it is an employee reply. If it equals 0, it is a user message.
   Request Method: GET
   Response Result:
   - Data for success:
     - Status Code: 200
     - Body
       - msg: “success”
       - data (array):
         - id
         - content
         - is_reply (“1” / “0”)
         - event_id
         - created_datetime (format:[YYYY-mm-dd HH:ii:ss], (The time that the message is created.)

- Data for event not found:
    - Status Code: 404
    - Body:
        - msg: “not found”

Body Result Sample for Success:

```
{
        “msg”: “success” ,
        “data”:  [
                {
                        “id”: 1,
                        “content”: “Lorem ipsum dolor sit.”,
                        “is_reply”: 0,
                        “event_id”: 1,
                        “created_datetime”: “2020-12-11 10:10:10”
                },
                {
                        “id”: 2,
                        “content”: “Lorem ipsum dolor sit.”,
                        “is_reply”: 1,
                        “event_id”: 1,
                        “created_datetime”: “2020-12-11 10:10:10”
                },
                ...
        ],
}
```

b. Add Message Under Specific Event (/message/{event_id})
Description: Send message under specific event. If ‘is_reply’ equals 1, it is an employee reply. If it equals 0, it is a user message.
Request Method: POST
Request Parameter:
- content
- is_reply (“1” / “0”)

Response Result:
- Data for success:
    - Status Code: 200
    - Body
        - msg: “success”
        - data:
            - id (message id)
            - content
            - is_reply
            - event_id
            - created_datetime (format:[YYYY-mm-dd HH:ii:ss])

- Data for wrong data:
    - Status Code: 422
    - Body:
        - msg: “data cannot be processed”

- Data for event not found:

- Status Code: 404
- Body:
    - msg: “not found”

Second 3 hours –Front End Development

Use the template provided in the material to develop the information management system for the enterprise. You should create all the functionalities for the page with communication with the backend web services API. You can modify the content of the API return prompt to improve the user interaction experience.

Staff panel

1. Authentication

Functionalities:

Login

- Input email and password to login.
- Alert for wrong response.

Logout

- Click the logout link, current user will be logged out and redirected to the login page.

2. Monitoring and management

a) Dashboard

- Current Time(Requested asynchronously and should be updated automatically)
- Unhandled Events List
  Each Unhandled event includes:
  - Event Code
  - Event Name
  - Event Date
  - Event Type
  - Event Status
  - Event Permission
  - Accept Button
- Handled Event List
  Each Handled event includes:
  - Event Code
  - Event Name
  - Event Date
  - Event Type
  - Event Status
  - Event Permission
  - Finish Button
- Unreplied Messages List
  Each Unreplied event includes:
  - Client
  - Event Name
  - Content
  - Reply Button

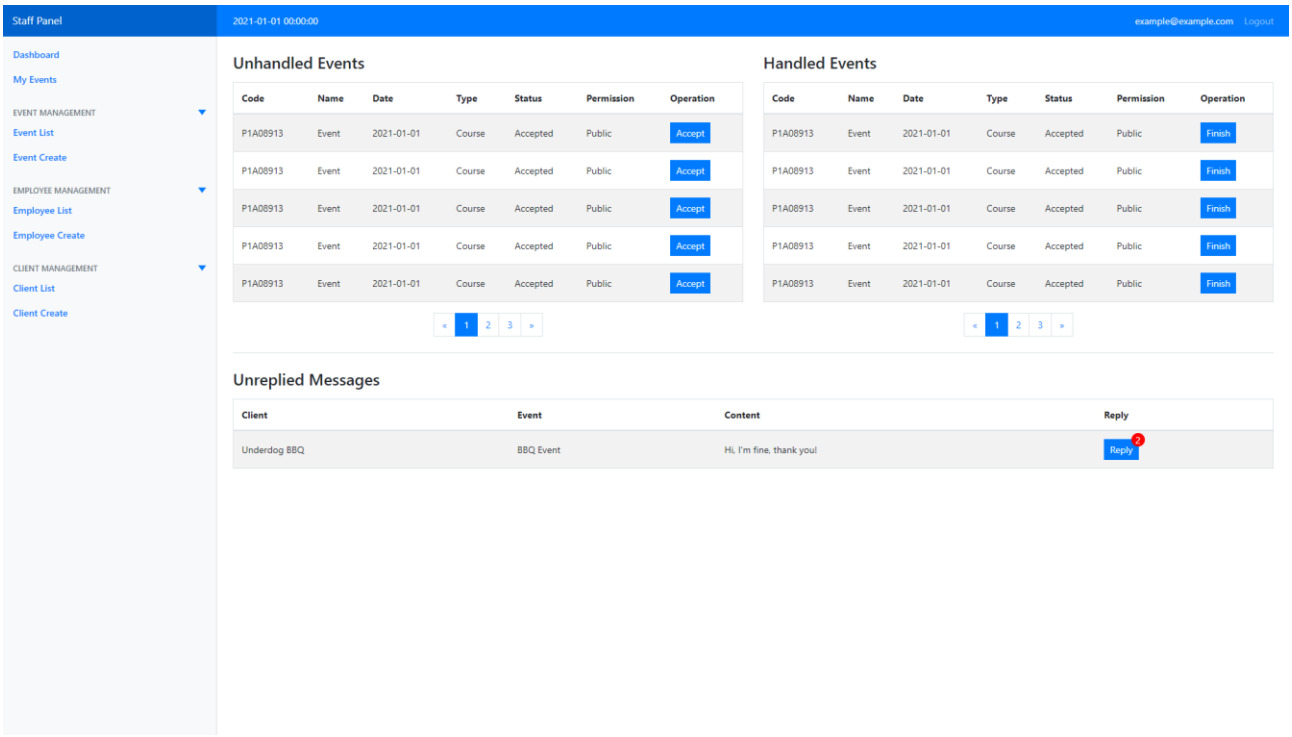
- When an employee (including admin) accepts an assigned event, the event moves from the Unhandled Events List to the Handled Event List.
- When all assigned employees in the event Click the accept button, the status of the event changes to accepted, otherwise the event status is still pending.
- When an assigned employee clicks the finish event button, the status of the event changes to finished.
- Only events with status accepted can be clicked to finish.
- In the Unhandled events list or handled events list, up to five events can be displayed at a time, and more than five events can be displayed by pagination.
- In the Unreplied messages List, each reply button displays the number of unreplied messages. Calculate the number of messages sent by user after the last reply from employee.
- Unreplied Messages List displays all unreplied events, The last message from the user is displayed in content.

b) My Event

Display all assigned(current account) events

Each event contains：

- Code
- Event Name
- Event Address
- Event Date
- Event Premission（Public、Private）



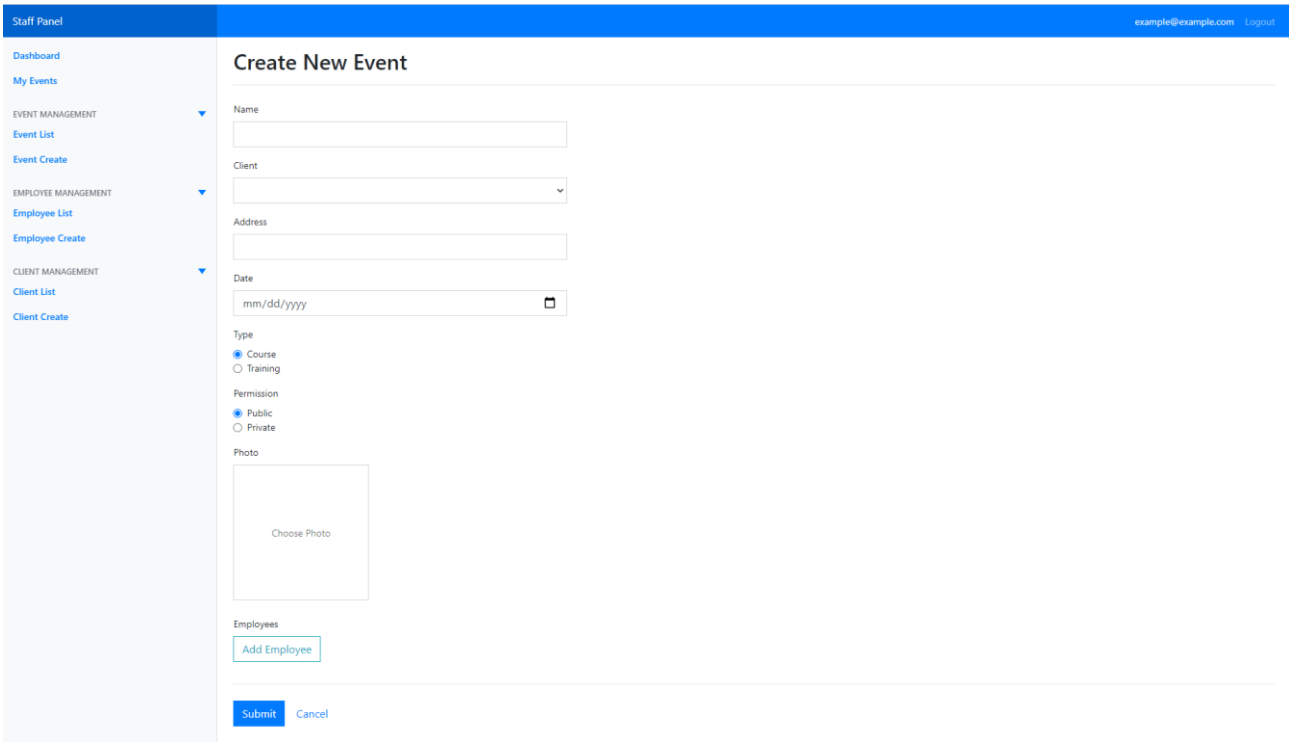
3. Event management

a) Create Event

- Event Name
- Client
- Event Address
- Event Date
- Event Type（Course、Training）
- Event Permission（Public、Private）

- Event Photo
- Employees
- Submit Button

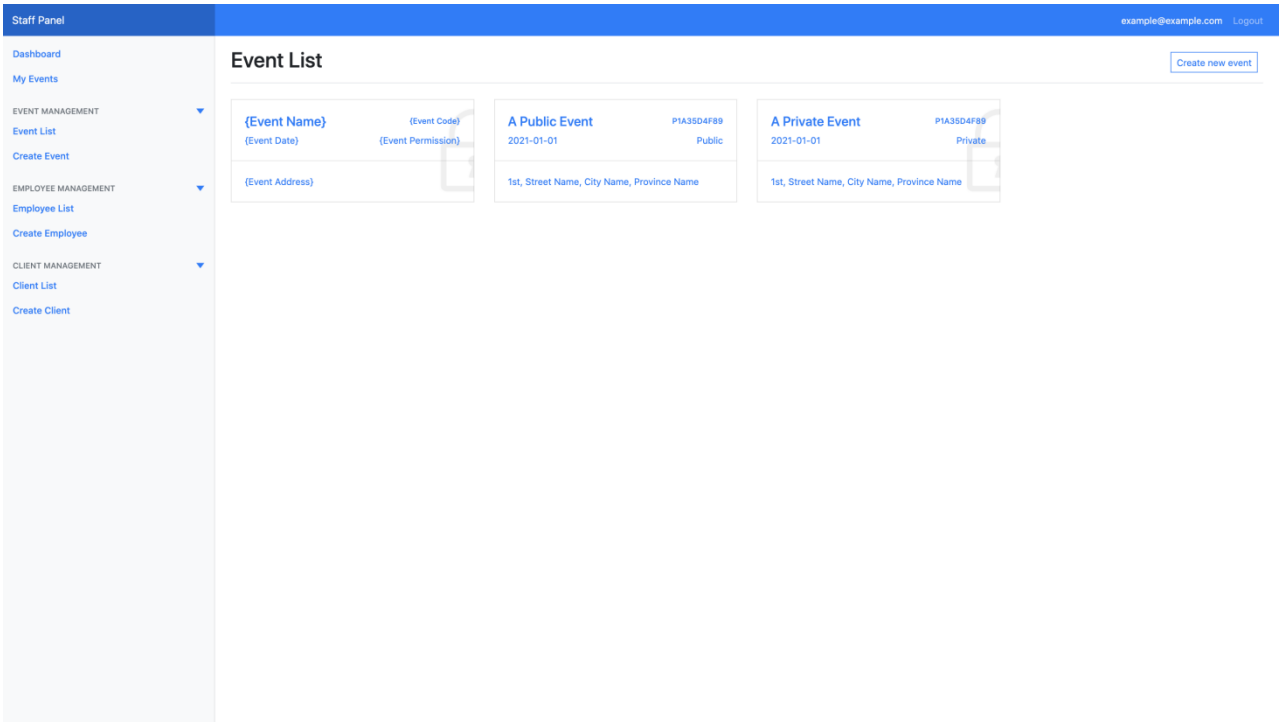
- After the event is submitted, a code with a length of 8 is automatically generated (globally unique, composed of letters and numbers, not case sensitive).
- The scope of the assigned employee must match the event's type.
- The Event Date should be a datepicker, and the next day is the earliest available date.



b) Event List

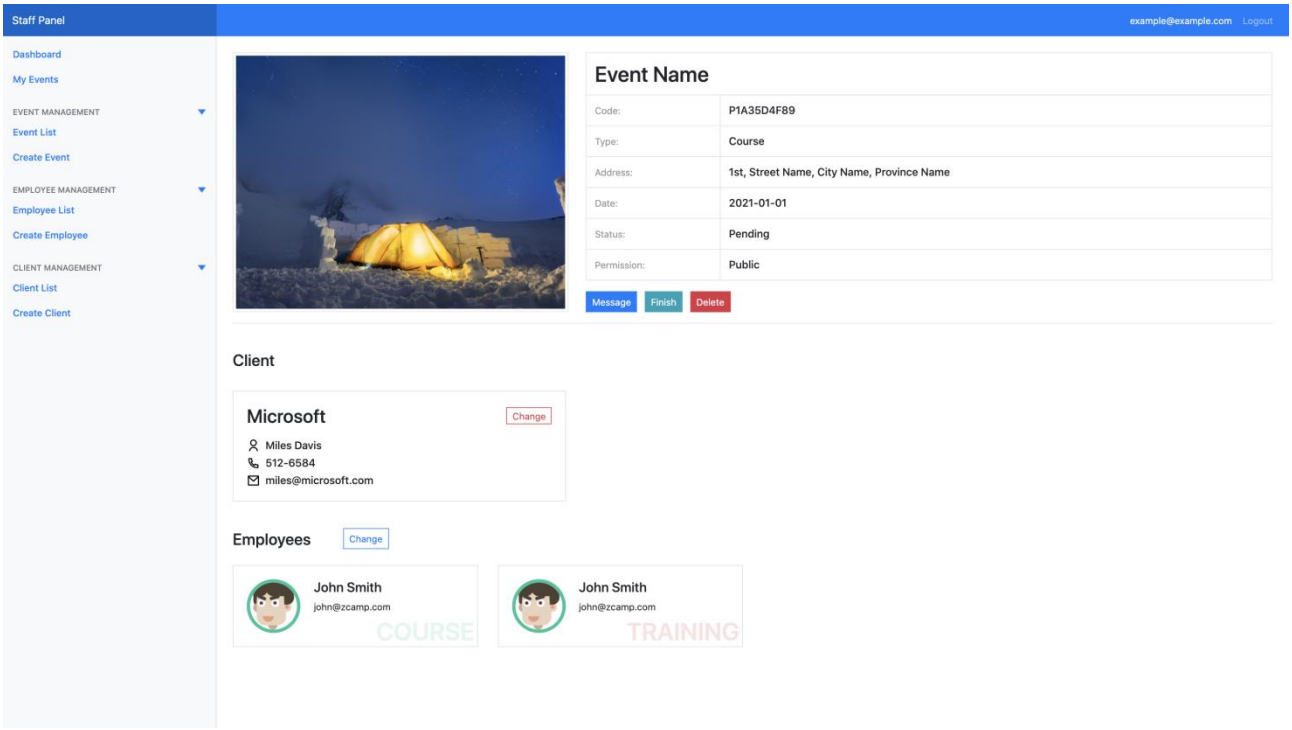
Each event contains：

- Code
- Event Name
- Event Address
- Event Date
- Event Premission（Public、Private）



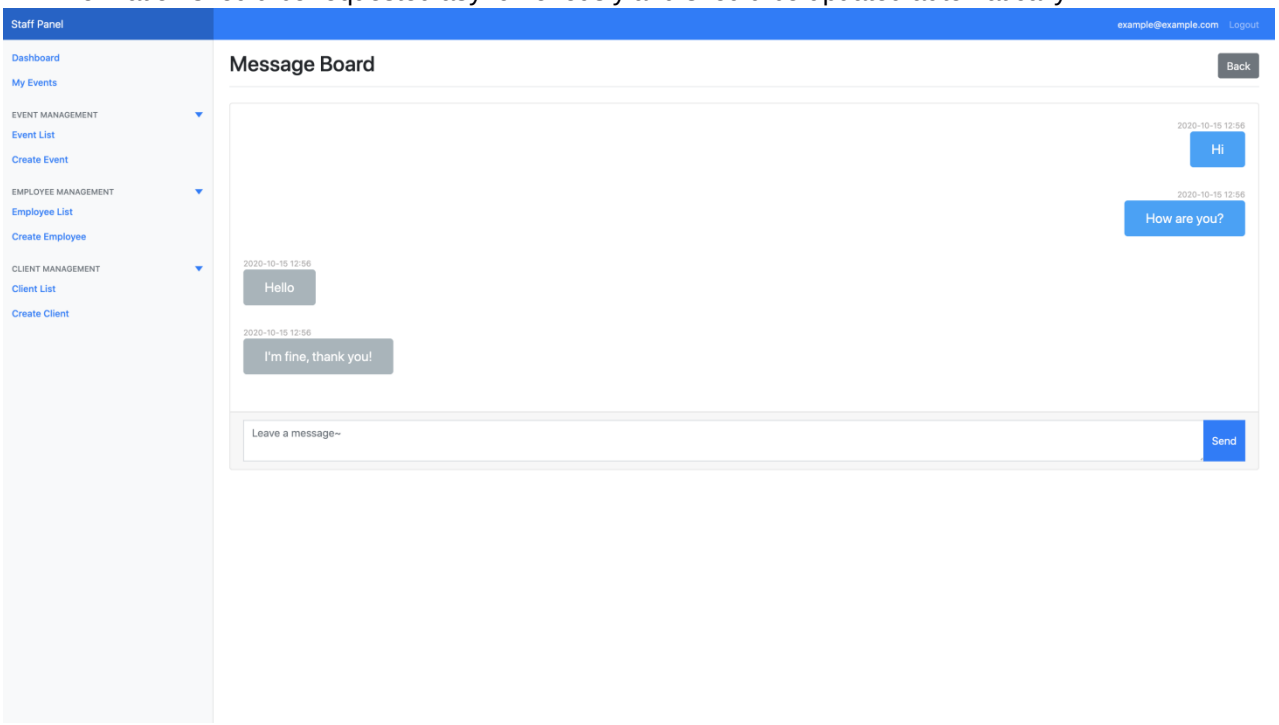
c) Event Detail
   - Event Name
   - Event Code
   - Event Type（Course、Training）
   - Event Photo
   - Event Address
   - Event Date
   - Event Status（Pending、Accepted、Finished）
   - Event Permission（Public、Private）
   - Client
   - Finish Button（Available,If event status is “Accepted”）
   - Message Button
   - Delete Button
   - Employees（Contains a change button to add or remove a new employees for admin only）

- The delete button can delete an event. After the event is deleted, the relevant enrollments and event information will not be displayed.
- Before deleting an event, you need to delete all employees in the event detail page.
- Event code is displayed in uppercase (case insensitive when used).
- Admin can finish an event, although it is not assigned to this event.
- When the event status is accepted, if the admin adds an employee or removes all employees, the event state should change back to pending.



d) Message Board
   - Message List
   - Input Box
   - Send Button

- Employees can view history messages and reply them in the message board. Employees can input the message content in the Input box and click the send button to submit the message.
- On the staff panel, the right side of the message list is the reply of the employee (including admin), and the left side of the message list is the message information sent by the user.
- Information should be requested asynchronously and should be updated automatically.

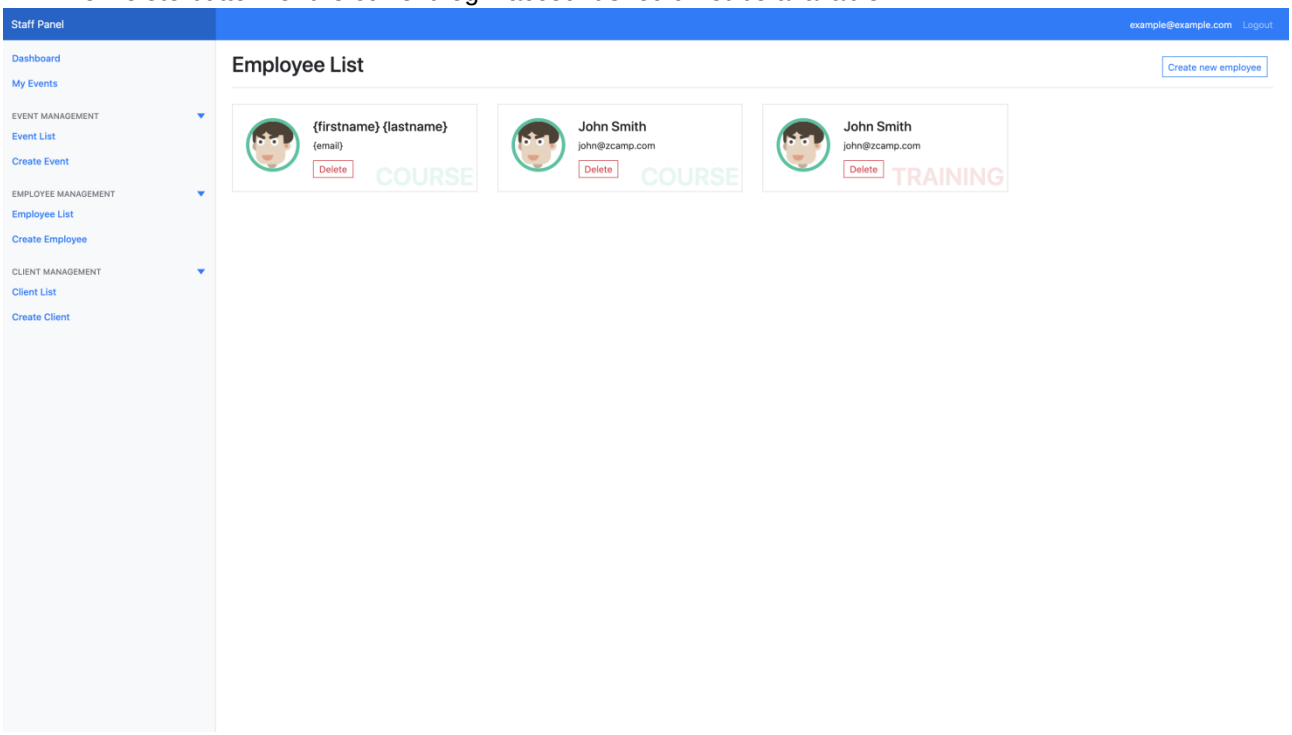


4. Employee management

a) Employee List

- Photo
- First Name
- Last Name
- Email
- Scope（Course、Training）

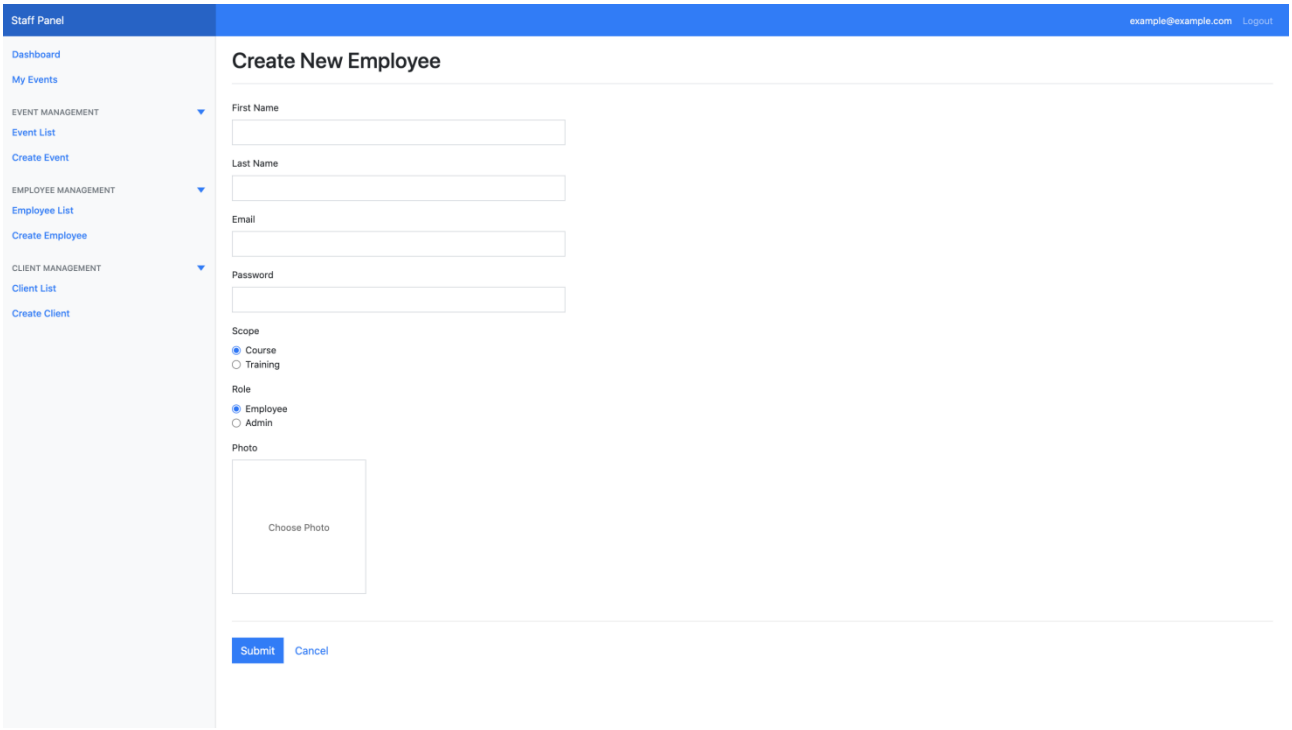
- Use the delete button to delete an employee. After deleting an employee, the employee will not be displayed in the related event.
- The Delete button for the current login account should not be available.



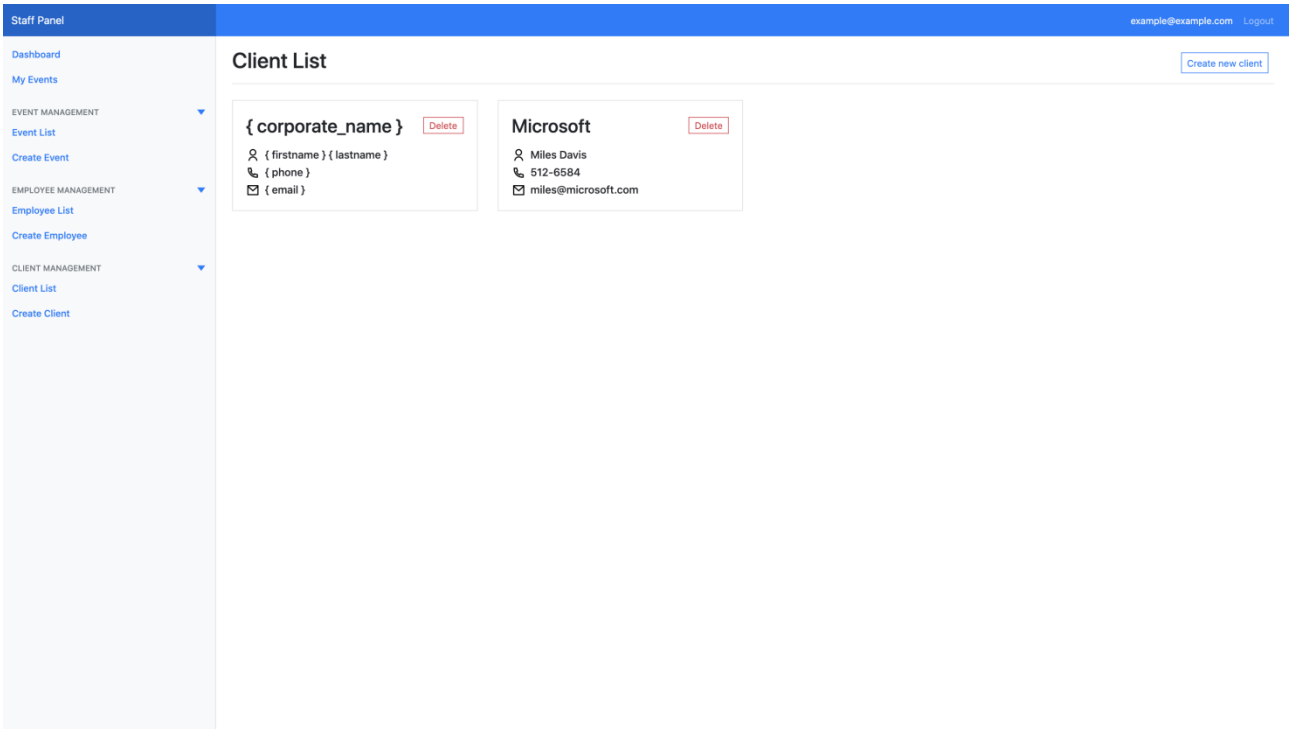
b) Create Employee

- First Name
- Last Name
- Email
- Password
- Photo
- Scope(Course、Training)
- Role(Employee, Admin)
- Submit button

- Click the submit button to add an employee. The email of the employee cannot be duplicate with other employees (except for the deleted employees).

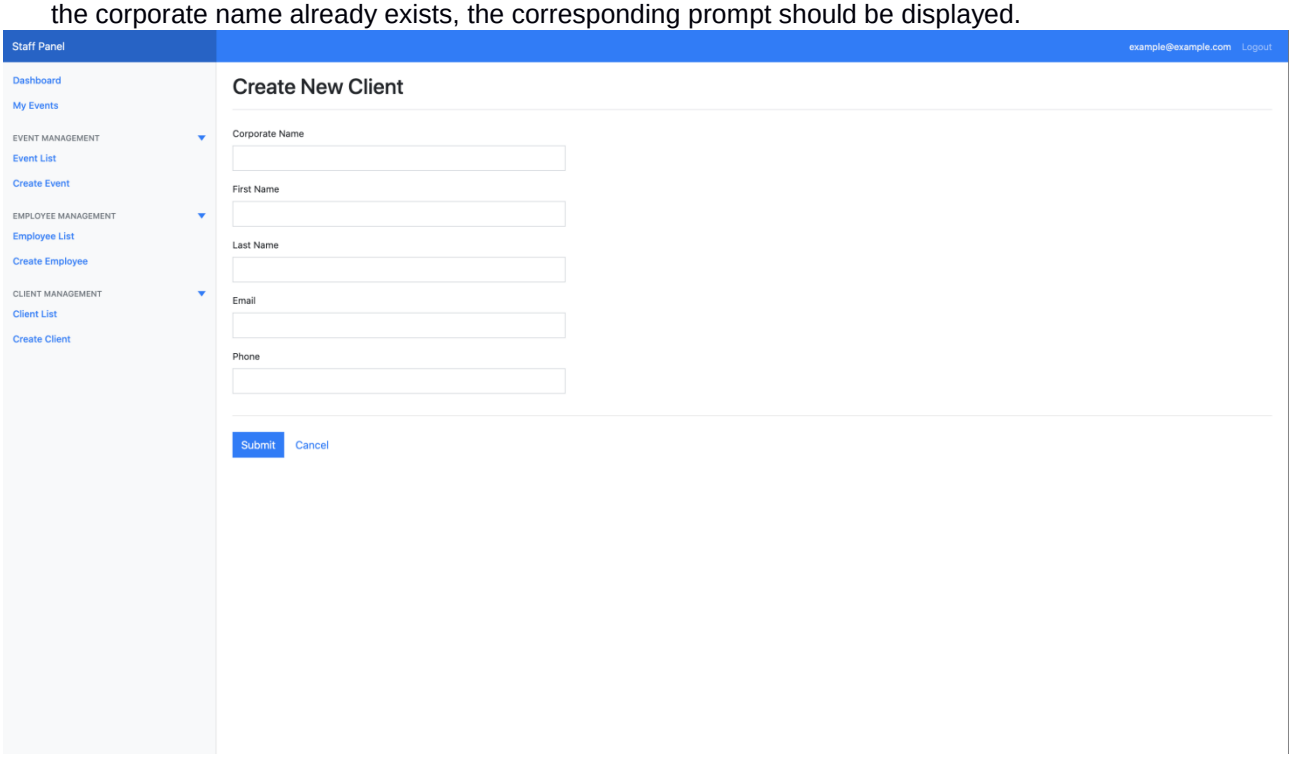


5. Client management
   a) Client List
      Each client contains:
      - Corporate Name
      - First Name
      - Last Name
      - Email
      - Phone
      - Delete button
   - Click the delete button to delete a client. (You need to delete all the events that belongs to this client before deleting it.)



b) Create Client
   - Corporate Name
   - First Name
   - Last Name
   - Email
   - Phone Number
   - Submit

- Click the submit button to create a new client. Two identical corporate names cannot be created. If the corporate name already exists, the corresponding prompt should be displayed.



User panel

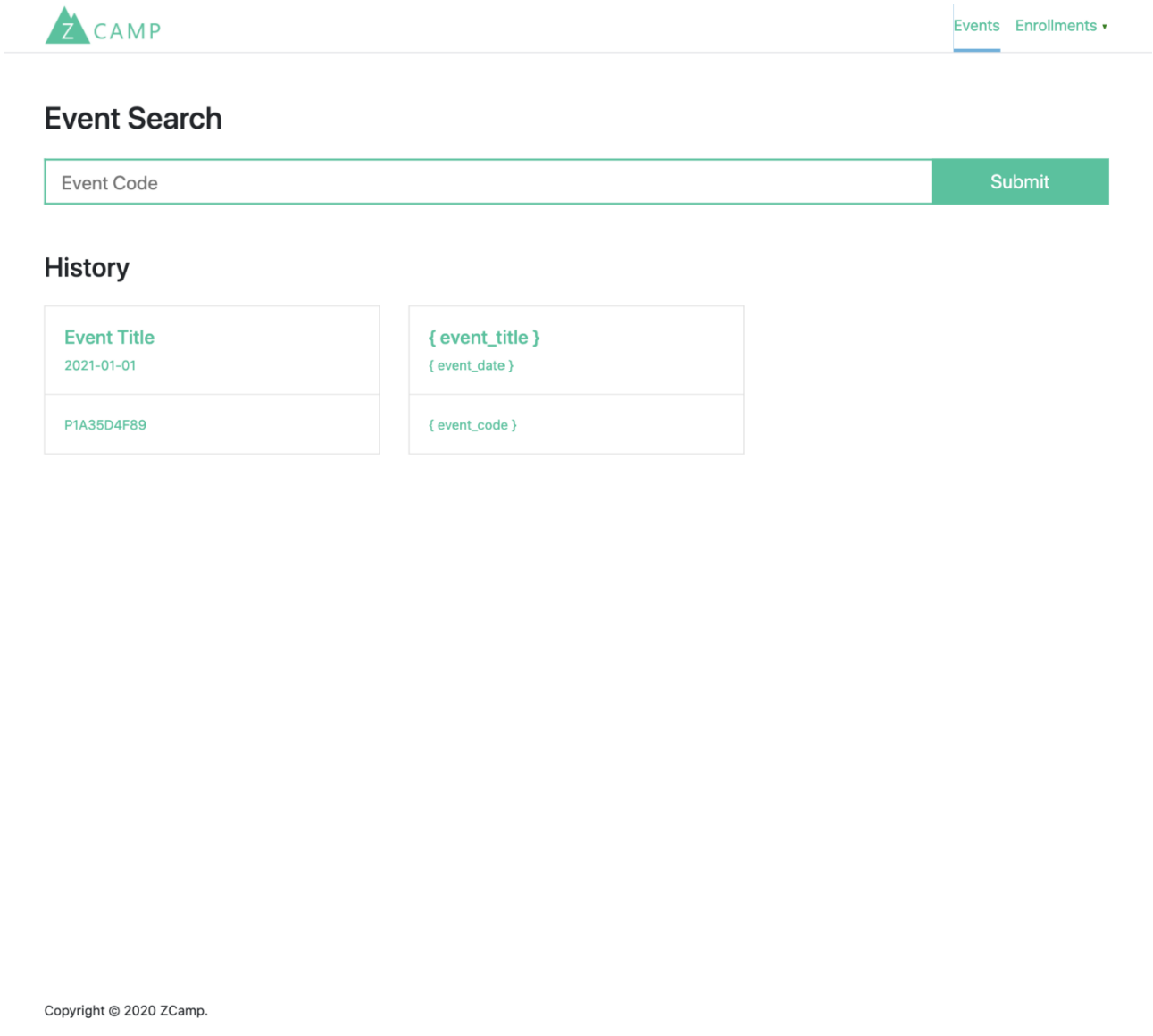
1. Event Management

a) Event Search

- Event Code
- Submit

History（Display the history of the last three searches）

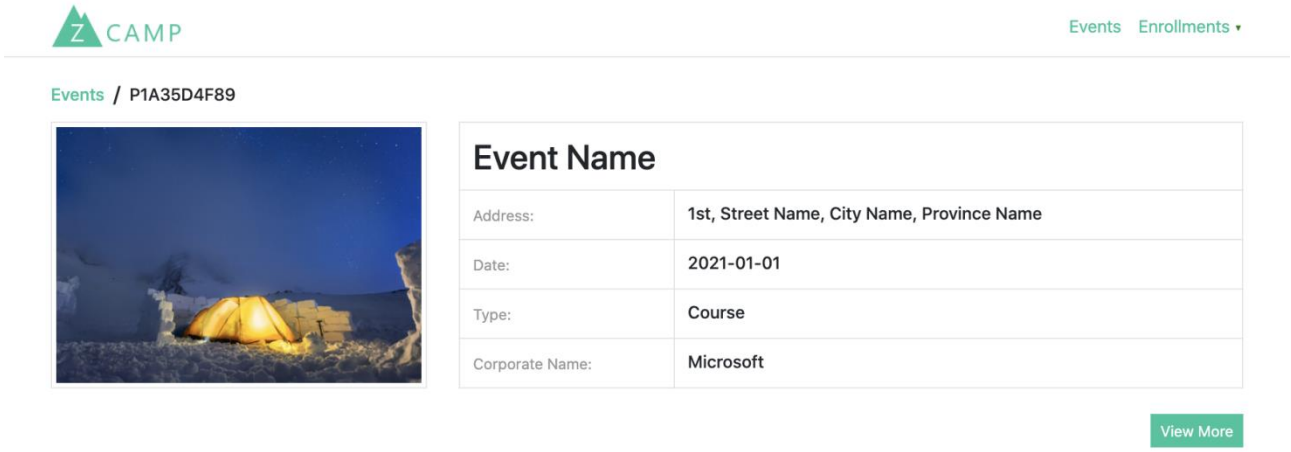
- Event Name
- Event Date
- Event Code



b) Event Preview

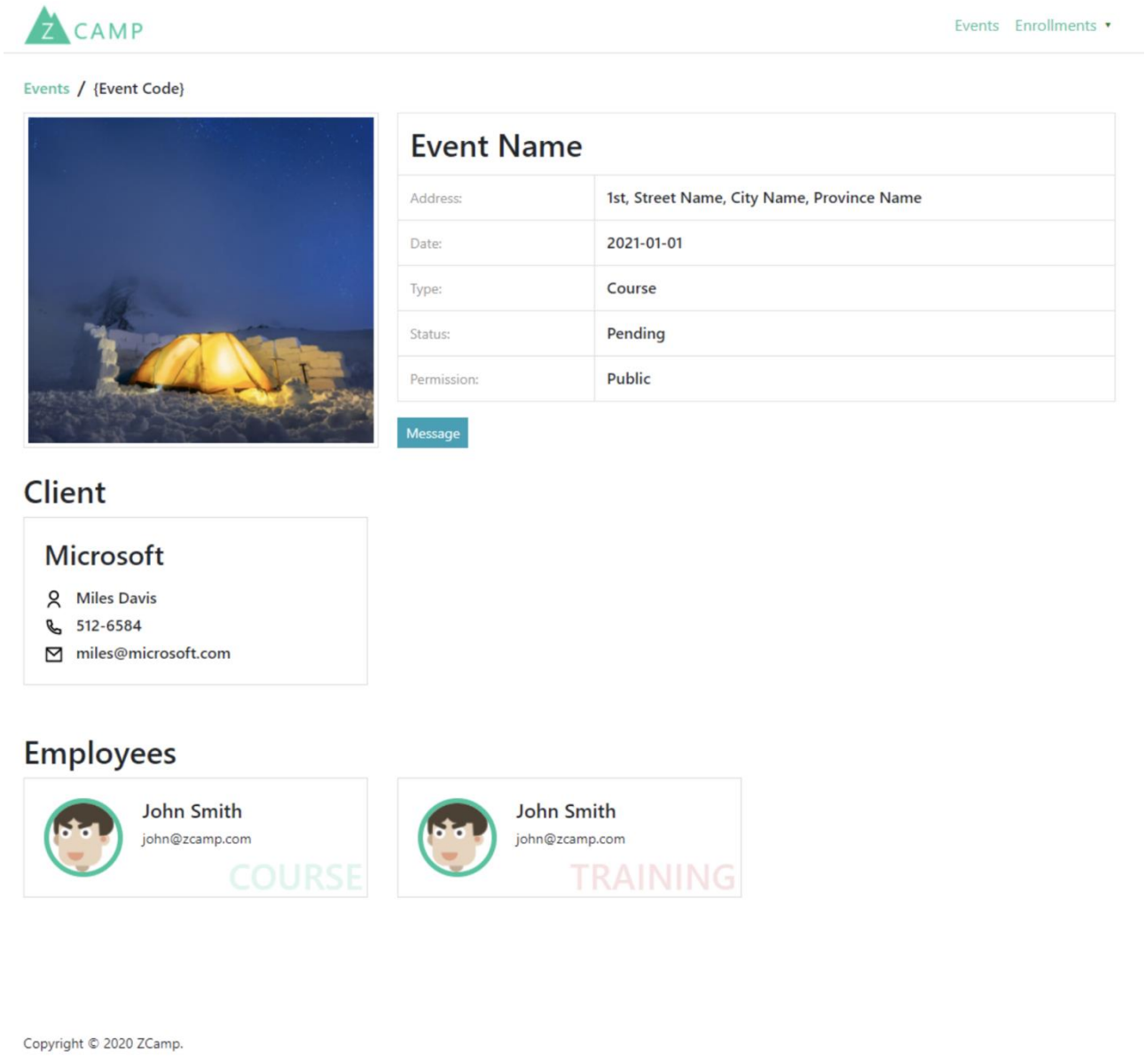
- Event Name
- Event Photo
- Event Address
- Event Date
- Corporate Name

- Event Type
- View More Button



c) Event Detail

- Event Name
- Event Photo
- Event Address
- Event Date
- Event Type
- Client
- Employees
- Event Status（Pending、Accepted、Finished）
- Message Button
- Permission



d) Message Board

- Message List
- Input Box
- Send Button

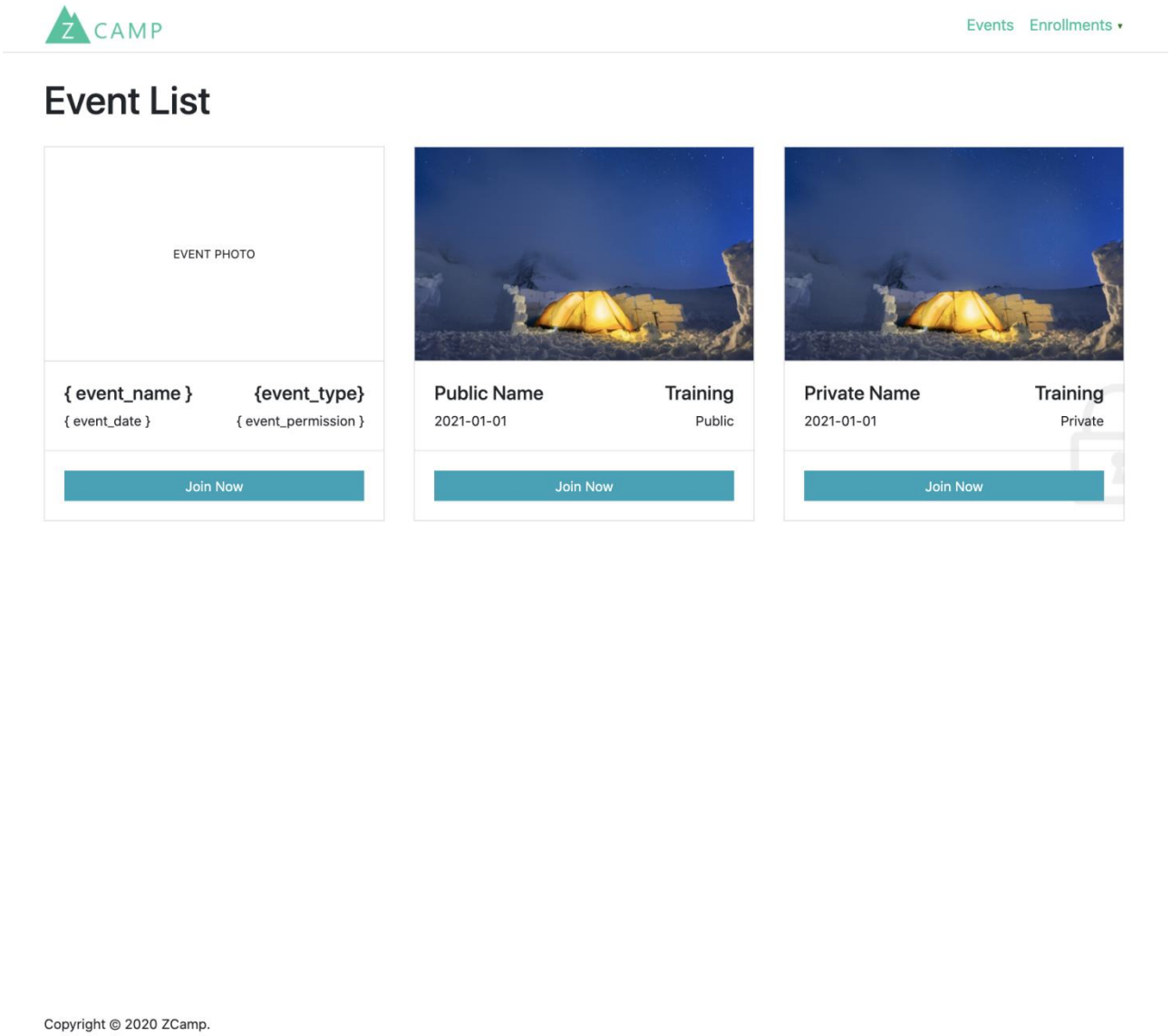
- In event management, the event search page should be displayed first. After the user enters the correct event code, the query results are displayed in the event preview page.
- After clicking view more on the event detail page, the user is required to provide the phone number of the client. After verification, the user will jump to the event detail page and display the relevant information.
- When the user clicks the message button on the event detail page, it will jump to the message board page. The event history message and employee response should be displayed in the message board. User can inputs the message content in the message box and click the send button to submit the message.

- On user panel, the left side of the message list is replies form employees, and right side is messages from user.
- Information should be requested asynchronously and should be updated automatically.

2. Enrollment Management

a) Event List

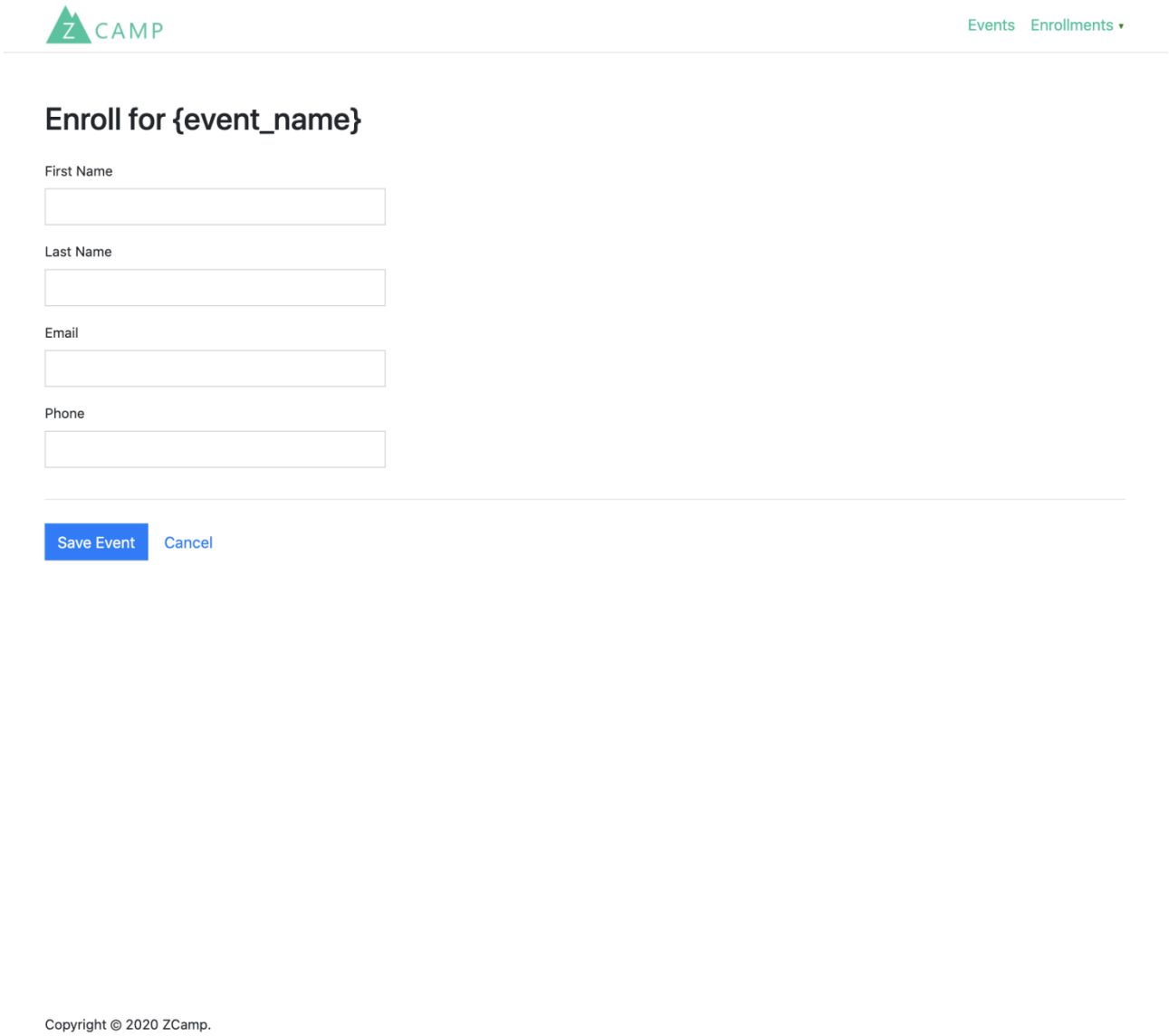
- Event Name
- Event Photo
- Event Date
- Event Type（Course、Training）
- Event Permission（Public、Private）
- Join Now



b) Event Enrollment

- First Name
- Last Name
- Email
- Phone

- Submit



- When the user clicks the join now button in the page, it will jump to the enroll page.
- If the joining event's permission is private, users need to enter the correct event code to jump to the enroll page. If the permission of the joining event is public, you can jump to the enroll page directly.

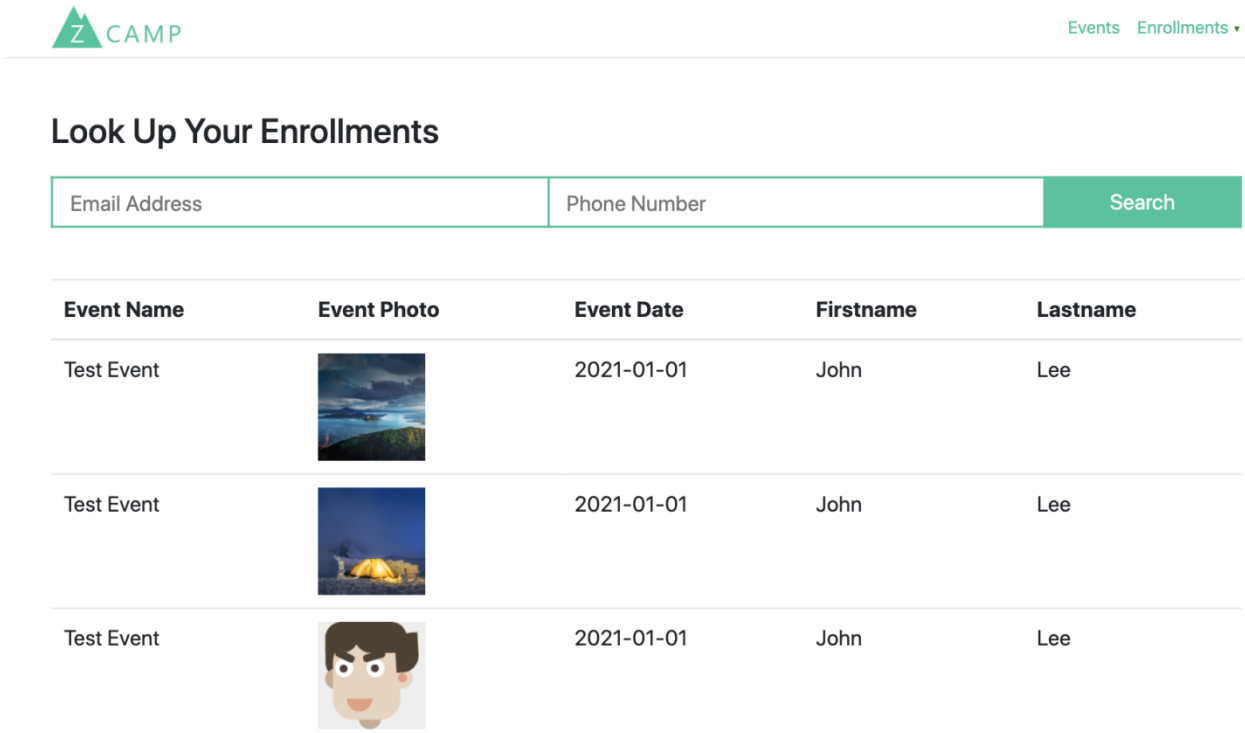
c) Event Enrollment History

- Email
- Phone Number
- Search button

Search Result

- Event Name
- Event Photo
- Event Date
- First Name
- Last Name

- When the user enters the correct email and phone number, click the search button, and all event records related will be displayed below the search box.
- When the input email and phone number cannot query the event enroll enrollments, the relevant info will be displayed on the page.



Additional requirement

- You need to create account for the project:
  - **Admin account**
    Account: admin@zcamp.com
    Password:admin
    Admin account should have full authority in the staff panel, At the same time, admin can also have a scope like an employee.

  - **Employee account**
    Account: employee@zcamp.com
    Password: employee
    Employee account can access staff panel, but can't access Event management, Employee management and Client management functions, employee cannot change assigned employees for an event.
- You should optimize the error content returned by the API so that users can understand the current problem.

- You can simplify the field names in the document in the process of developing front-end pages. For example, event name can be simplified to name. However, the simplified field name should keep the same meaning as the original field name, otherwise it will be considered as missing the required field.

Instructions to the Competitor

1. The media files are available in the ZIP file. You need to use the template provided in the material to make the front-end function, allowing some modifications to improve the interactive experience, but should not remove the existing fields. You can use any supplied JavaScript and PHP framework if you find it necessary.
First 3 hours: Files to be collected after the first phase on the server:

- Web service (http://<hostname>/XX_Module_A/api/v1/..)
- Folder inside webroot (XX_Module_A/)
- ERD screenshot named “XX_ERD.png” in “db-dump” folder inside of XX_Module_A
- Database dump named “XX_database.sql” in “db-dump” folder inside of XX_Module_A

Second 3 hours: Files to be collected after the second phase on the server:

- Web service (http://<hostname>/XX_Module_B)
- Staff Panel webroot (http://<hostname>/XX_Module_B/staffpanel)
- User Panel webroot (http://<hostname>/XX_Module_B/userpanel)

FILES PROVIDED

ITEM	DESCRIPTION
Media files	Template files

INTERNET ACCESS

- no internet access