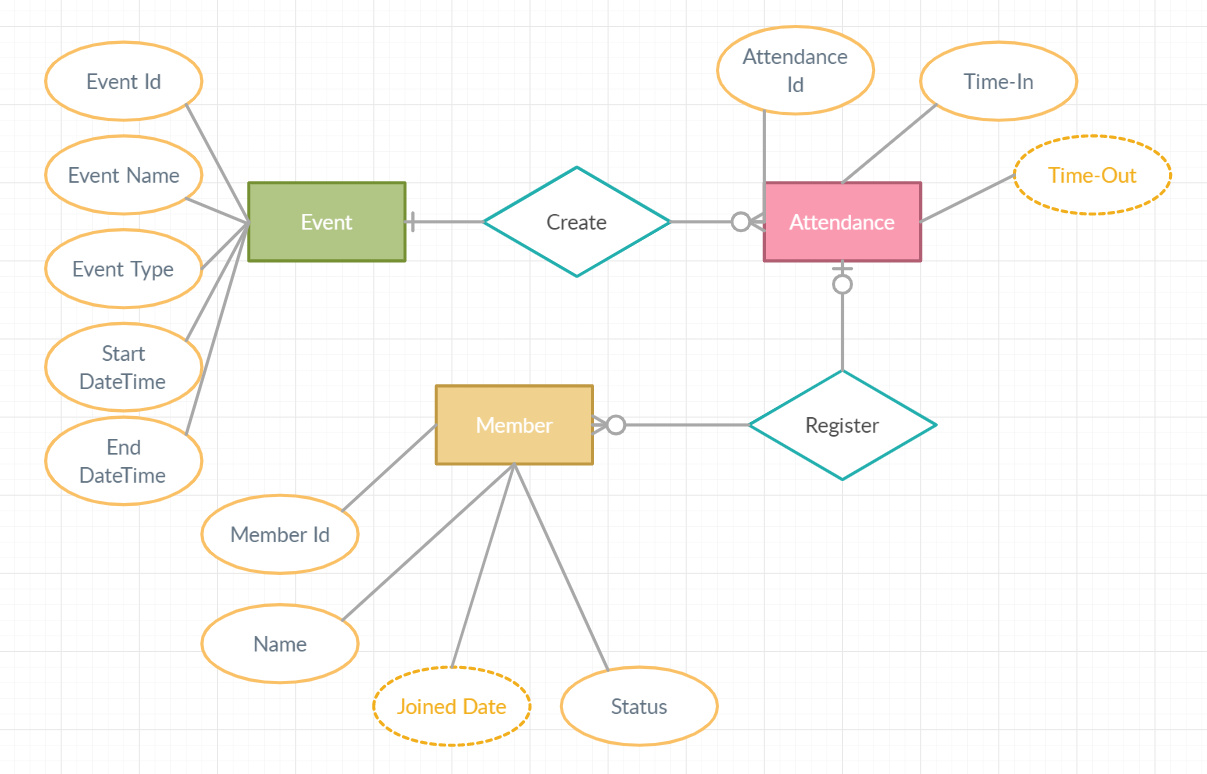
**Node JS Masters Final Project**

Congratulations for completing the lectures for NodeJs masters. It’s now time to put those stuff we learned into test by creating your own backend using NodeJs, Express and MongoDB. Please refer below for the requirements for your final project.

**Project Overview**

Create a NodeJS backend for attendance monitoring for a small event. Implement this using REST-ful API that will support CRUD operations for Events, Participants and Attendance. Refer to ERD diagram below for entity relationship and information.



Note: Dotted property means optional field

**Business Requirements**

**Events Endpoint**

* Resource: /api/events/
* **GET**: /events/
  + Return all Events
* **GET**: /events/Id
  + Return an event by EventId
  + Return Event object with array of **MemberAttendance** object
    - MemberAttendance
      * MemberId (GUID)
      * Name
      * TimeIn
      * TimeOut
* **POST:** /events/ and **PUT** /events/
  + Accept Event object
  + Event start date should be < event end date
  + Required fields validation check
* **DELETE:** /events/
  + Return a validation error if there is an event attendance
* **GET**: /events/search?eventname=[searchEventName]&datestart=[searchDateStart]&dataend=[searchDateEnd]
  + Search events by Event Name, DateTime Start, DateTime End
  + All fields are optional
  + Will return an error if no search criteria provided
* GET: /events/export?eventId
  + Return an excel file with the following details based on EventId
    - Filename: [EventName]\_[EventStartDateTime].xlsx
    - Columns
      * Member Name
      * Time-In
      * Time-Out
    - Sort results by Time-In, Asc

**Members Endpoint**

* Resource: /api/members/
* **GET**: /members/
  + Return all Members
* **GET**: /members/Id
  + Return a member by MemberId
  + Return Member object with array of **EventAttendance** object
    - EventAttendance
      * EventName
      * TimeIn
      * TimeOut
* **POST:** / members / and **PUT** / members /
  + Accept member object
  + Required fields validation check
* **DELETE:** /members/
  + Return a validation error if there is an event attendance
* **GET**: /members/search?name=&status
  + Search member by Name and status
  + Status are enumerations of
    - Active
    - In-active

**Attendance Endpoint**

* Resource: /api/attendance/
* **POST:** /attendance/ and **PUT** /attendance/
  + Accept Attendance object
  + Time-in date should be < Time-out date
  + Required fields validation check
* **DELETE:** /attendance/
  + Delete record

**Technical Approach**

The following requirement must be followed when building your NodeJs backend.

* Use Express.js for creating your RESTful endpoint
* Use middleware for validation logic
  + You can use any 3rd party middleware for validation check
* When error occurred or validation check fails, the API should return a JSON with the following properties
  + Status (400, 404, etc)
  + Result (Validation Error, Internal Error)
  + Validation Messages (the fields that fails validation)
* Separate your logic between routers, controller and service
* Avoid declaring variables using “var”
* Export your Postman API Calls/Rest-client file and include it on your repository
* Put your codes on your Git repositories and send the Git repository URL when submitting your work
* Use LowDB as database
* You can also use MongoDB/mongoose deployed cloud (+15 on final project)
* Add logging on each end-point call and implement it using Event Emitter
  + Log file should be on per-day basis (if there is a log file exist for today, update it, else, create new log file)
  + Log file name will be “AttendanceMonitoringLogs-[yyyy-mm-dd]”
    - Where date is the current date
  + Each API call should log the endpoint being called and the request parameters/body contents

**Tips/Suggestions**

* For middleware validation library, you may use:
  + <https://github.com/sideway/joi>
  + <https://express-validator.github.io/docs/>
* For document schema relationship, you may use: <https://mongoosejs.com/docs/populate.html>
* For Express app error handling, you may follow this approach: <https://expressjs.com/en/guide/error-handling.html>
* For Express resource downloading, you can use: <http://expressjs.com/en/5x/api.html#res.download>

Deadline of submission: Apr 5, **2021**