**Guest Lecture Management System**

**Software Engineering Project**

**Problem Statement –**

There are many speakers and events being held in colleges and most of the time communication is done either through WhatsApp or word of mouth or by mail and there are chances that the target students would miss this and there is a lack of a centralized medium where students can access everything related to a particular event from Resources to details about the speaker and the event. Hence, we created a Guest Lecture Management System which would fill this void and fulfil this requirement. While creating this project, we ensured that nothing is stored locally and made use of cloud services to ensure easy scalability and deployment if needed.

**Tech Stack Used –**

**FrontEnd** – ReactJS, Tailwind CSS

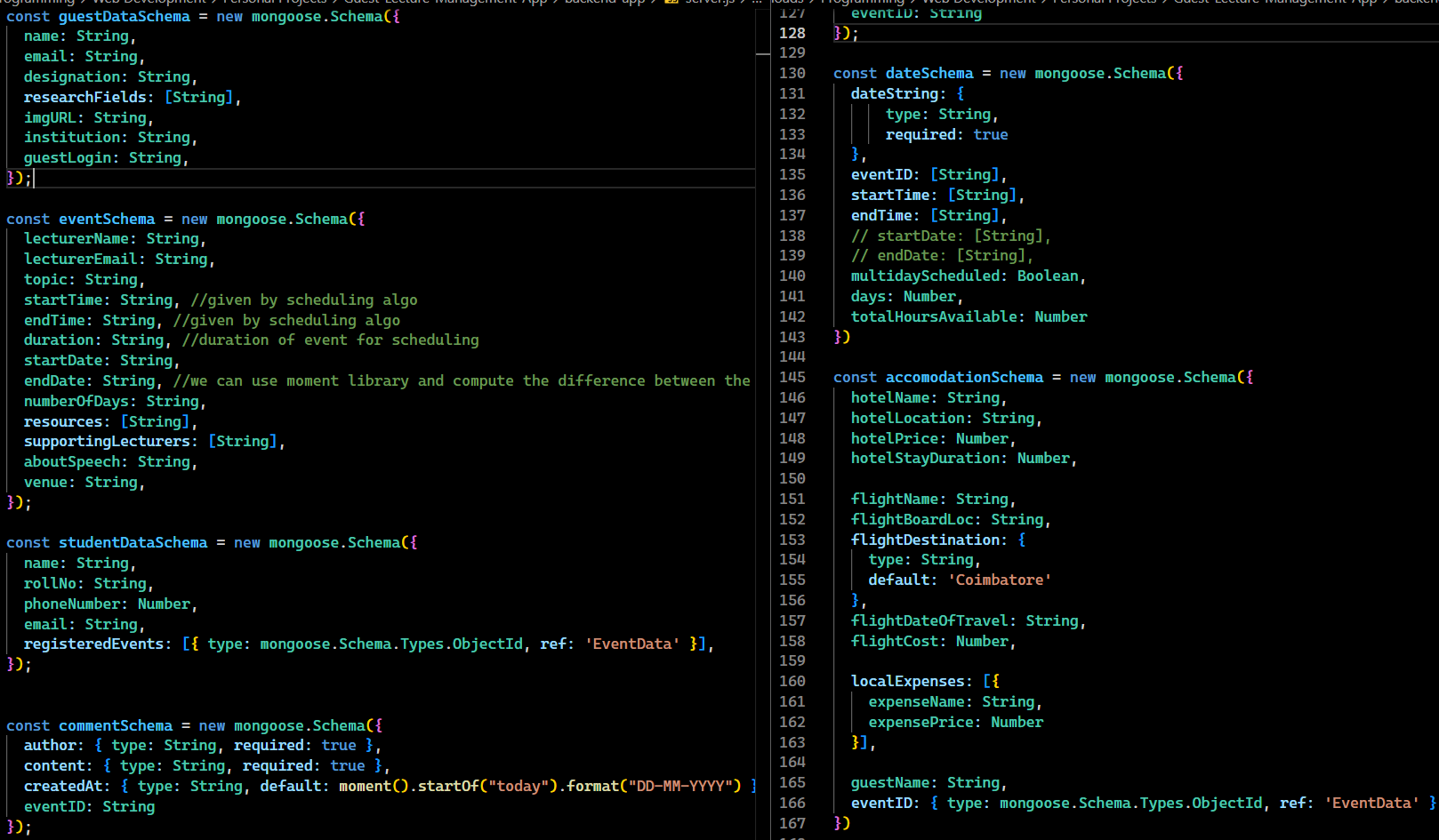
**Backend** – NodeJS (Express Server, Multer for processing files, Mongoose for MongoDB interfacing, AWS-SDK for accessing S3 Buckets, pdfmake for creating PDFs dynamically with data, Amadeus to get details about nearby hotels and flights to book for the guest, Nodemailer for sending emails related to the event and cors for enabling communication between the frontend and backend running on different ports on the same machine), Firebase(BaaS – Backend-As-A-Service) Authentication(OTP Authentication, Email ID login)

**Database** - AWS S3 Storage, MongoDB ATLAS (NoSQL Document-based database)

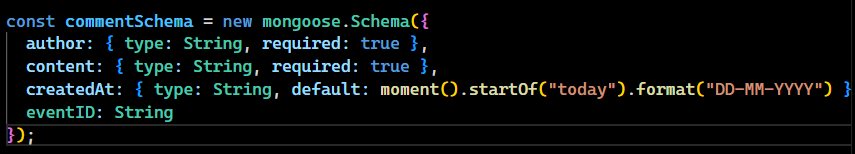
**Elaborating Backend Section of the Project –**

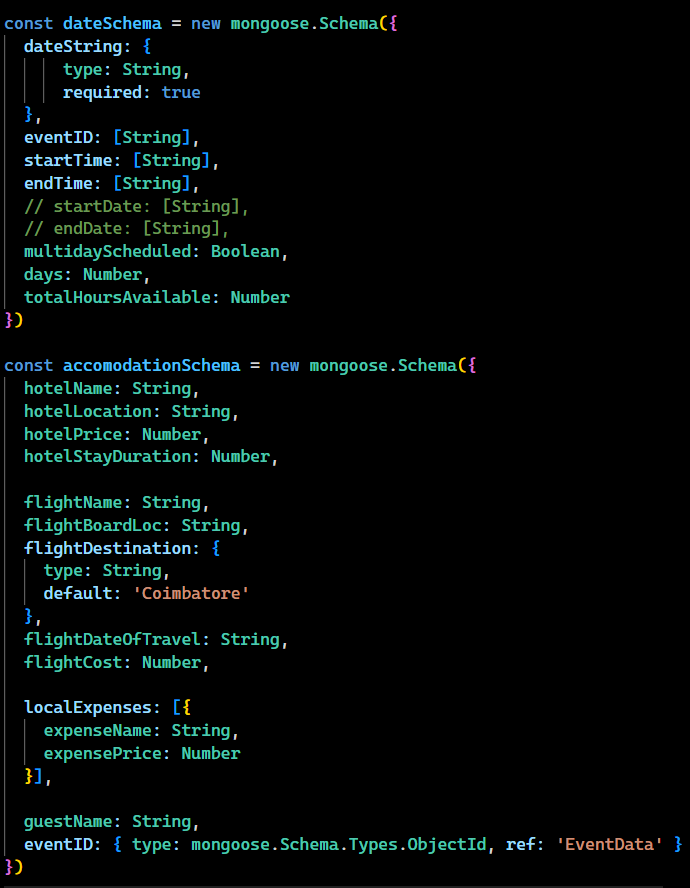
1. MongoDB Schemas:

* **GuestDataSchema** – Core document used to store the details about the guest lecturer
* **EventSchema** – Details about a particular event where the lectureName and lectureEmail are considered as the link between GuestDataSchema and the EventSchema. We figured that this pair of lectureName and lectureEmail would be globally unique for everyone.
* **StudentDataSchema** – This would contain the details about each student along with an array of events registered by that student. This is referred from the eventSchema.
* **CommentSchema** – To handle and store the Q&A section which has details like date, author and content.
* **DateSchema** – The schema needed for our Scheduling algorithm which automatically schedules an event to the next available slot. [**Scheduling Algorithm –** **It takes in fields like duration of the event and the number of days that event will be there for. We have also kept a particular day to have events only from 9am till 5pm. We have stored the current free slot on a particular day; hence the algorithm will check if that slot added with the duration of the new event is feasible for that day and only if it is possible will the event get scheduled on that day, else it will go to the next day and repeat the process. In the case of multi-day events, we ensure that the next couple of days have the slots free during the same window and only then will the event get scheduled, else the algorithm keeps moving forward till it finds a date that is feasible with this.**]
* **AccommodationSchema** – This document holds the details of a guest lecturer’s accommodation like Flight details, Hotel Details, local expenses.



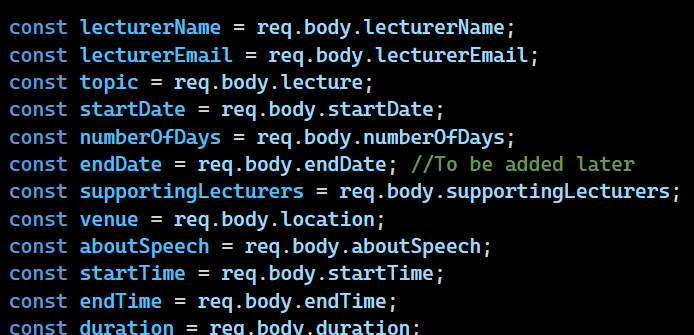
//we can use moment library and compute the difference between the dates





1. HTTP Requests
   1. GET Methods
      1. **/api/comments/:eventID** – Fetches all the Comments in the discussion forum of a particular event. This call accesses the CommentSchema
      2. **/api/check-student-profile/:email** – Check if a particular student is already registered or has to register for the first time. This API is called when they log in and checks the studentDataSchema for the presence of the email passed.
      3. **/api/list-lecturers** – Get the list of Lecturers along with their Email. This is used to show a dropdown of all past lecturers when creating a new event. It would be easier to reuse the data of already visited lecturers without entering all the data from scratch.
      4. **/hotelAndFlight** – Get the details of the hotel and flight which has been booked.
      5. **/api/getFiles/:eventID** – Get all the files uploaded by a lecturer for a particular event so that the student can view and access any particular file.
      6. **/api/guestDetails/:name/:email/:topic** – Get details about a particular lecturer along with the topic they’ll be presenting. Accesses the GuestDataSchema
      7. **/api/events/:email** – Get details about a particular event hosted by a lecturer. The lecturer’s email is used as the connecting link between GuestDataSchema and EventSchema.
      8. **/api/getAccomodation** – Get details about a particular guest’s accommodation. Accesses the accommodation schema for the same.
      9. **/api/checkRegistered/:email** – Check if a student has already registered for an event or not.
      10. **/api/checkPhoneNumber/:phoneNumber** – Check if the phone number entered by a student is valid or not part of the database. Used for OTP Login.
      11. **/api/get-student-data/:email** – Get details about a student.
   2. POST Methods
      1. **/api/comments** – Post request made to CommentSchema to insert a new comment under the discussion page of an event
      2. **/api/add-student-profile** – Create a new student profile when they register for the first time. This POST Request accesses the StudentDataSchema
      3. **/api/addEvent** – This would create a new event with the various details entered by the admin.

NOTE – The scheduling algorithm would be called immediately after the admin enters the event details and this method would receive the details after the scheduling algorithm. The eventSchema would first be populated and then the lecture details would be fetched to populate the accommodation details. And finally, the date details would be updated so that the next time an event is scheduled, the scheduling algorithm would take the updated date details. (you can add additional details in case I missed anything)



* + 1. **/api/scheduler** – The main scheduling algorithm. (already explained above)
    2. **/api/uploadFile** – Upload reference materials and resources which are used by the guest lecturer during the event. The guest lecturer can upload the files which can then be viewed by the students. Multer and AWS S3 is used here to upload the file.
    3. **/api/openFile –** Once a student clicks on a file, the signedURL is got from AWS S3 and this is used to view or download the file uploaded by the guest lecturer. The URL is made to have a ttl of 1 hour for security reasons and not abuse the S3 file access.
    4. **/api/add-guest-lecturers –** In case a new lecturer is coming for an event, this will add the details about that particular lecturer. This accesses the GuestDataSchema to store the details. The lecturer’s image is also got and this is handled using Multer.
    5. **/api/registerEvent/:eventID/:email** – This API call is made when a student wants to register for a particular event. After successful registration, an email is sent to the student’s mail ID about the same which contains the details about the event and the studentDataSchema is updated to have the current event appended to registeredEvents.
    6. **/api/generateReport/:type/:value** – Generate a PDF about the expenses incurred. This is done using a node-module called pdfmake.
    7. **/api/deleteEvent/:eventID** – An admin can delete a particular event in case it has been cancelled for unforeseen circumstances.