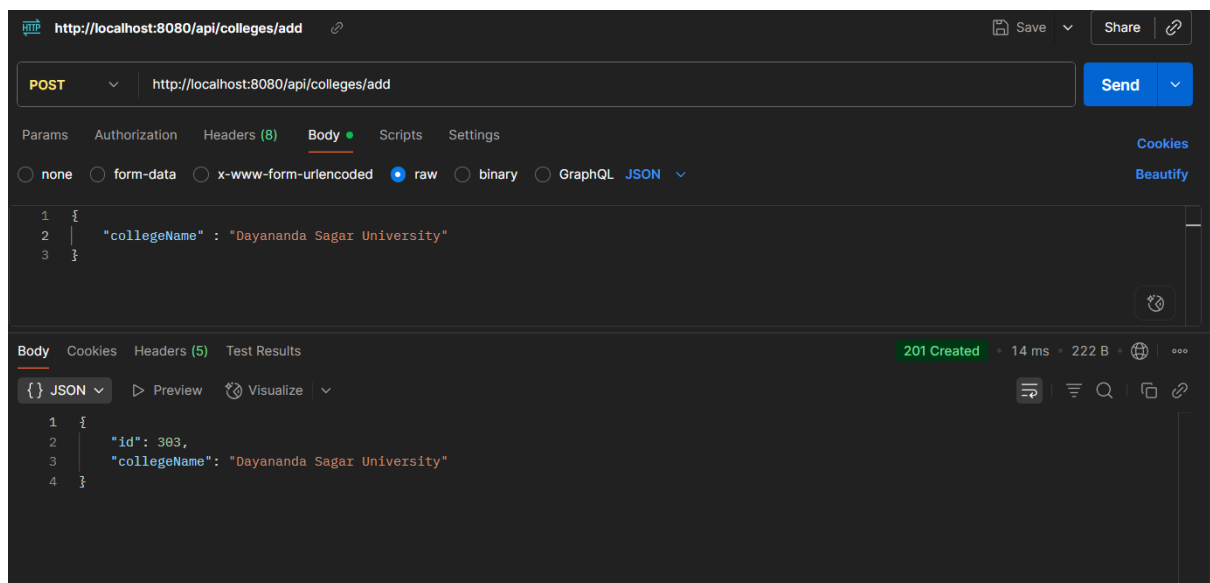
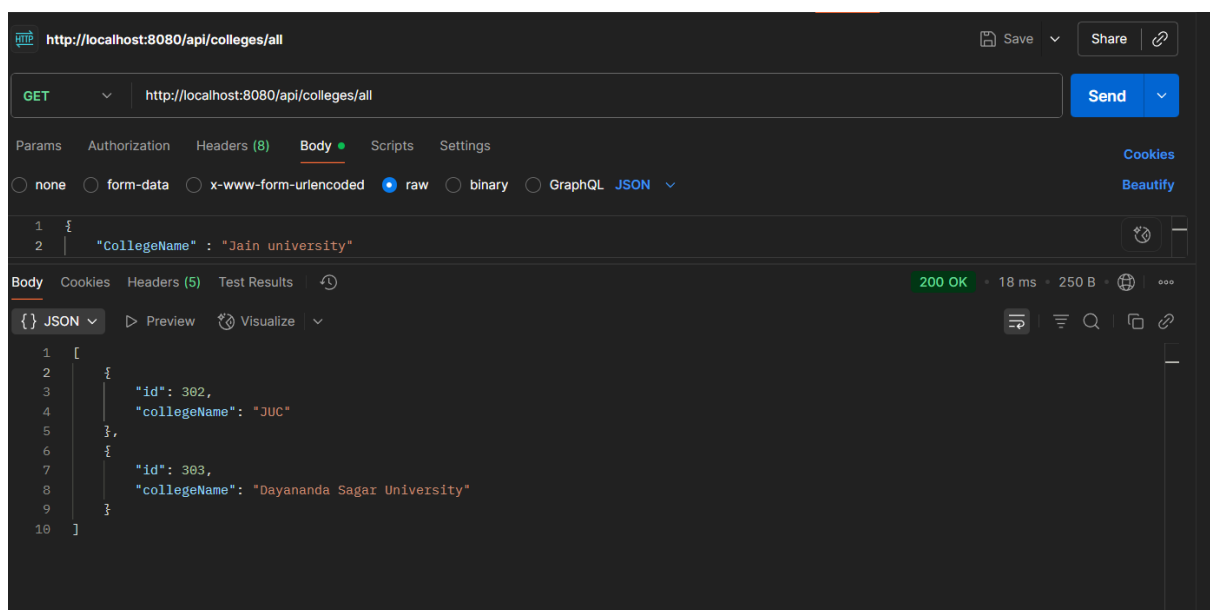


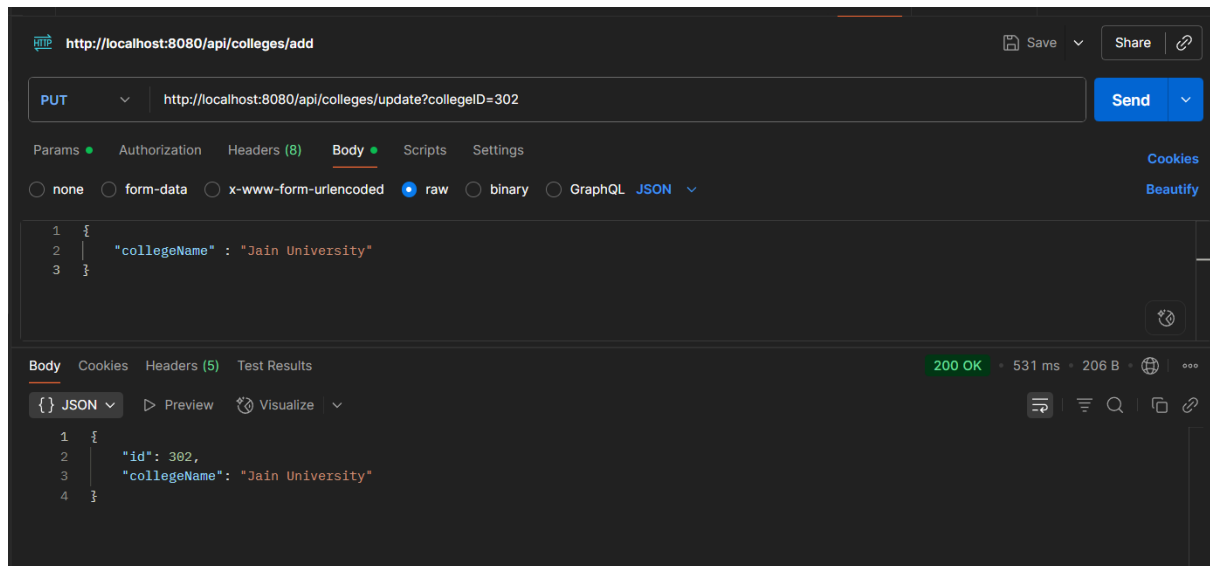
College And Event API Endpoints:



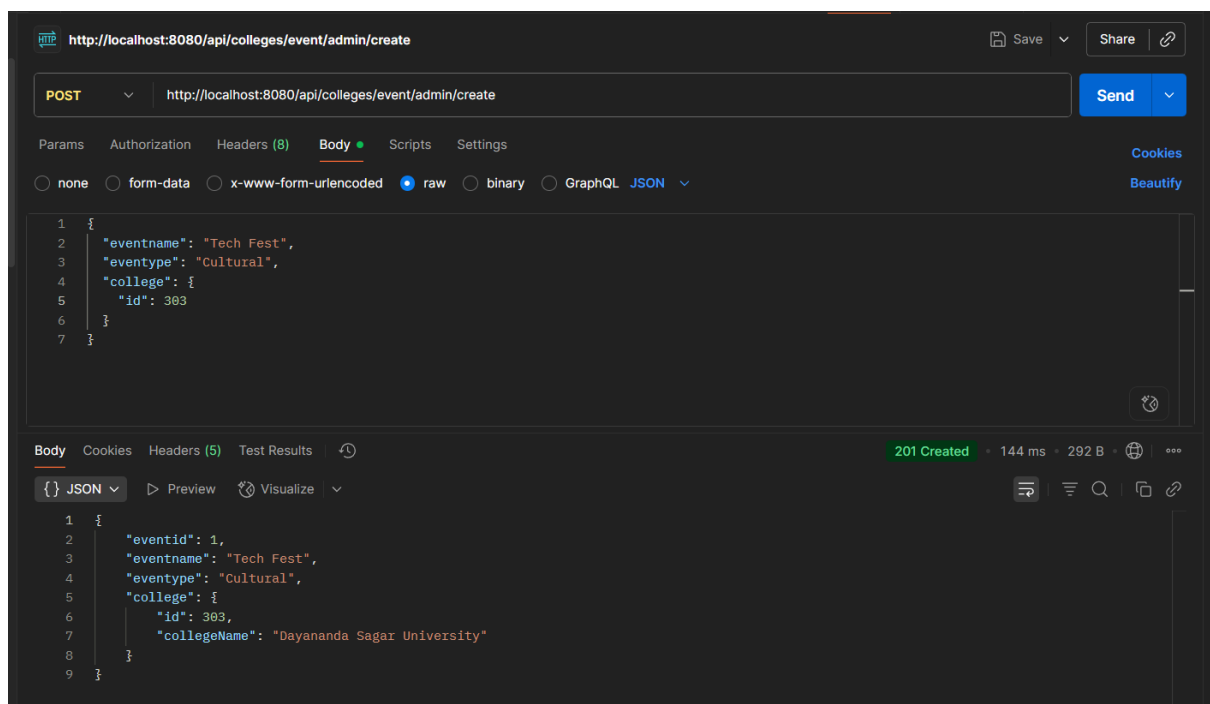
POST method to add colleges with an unique college id



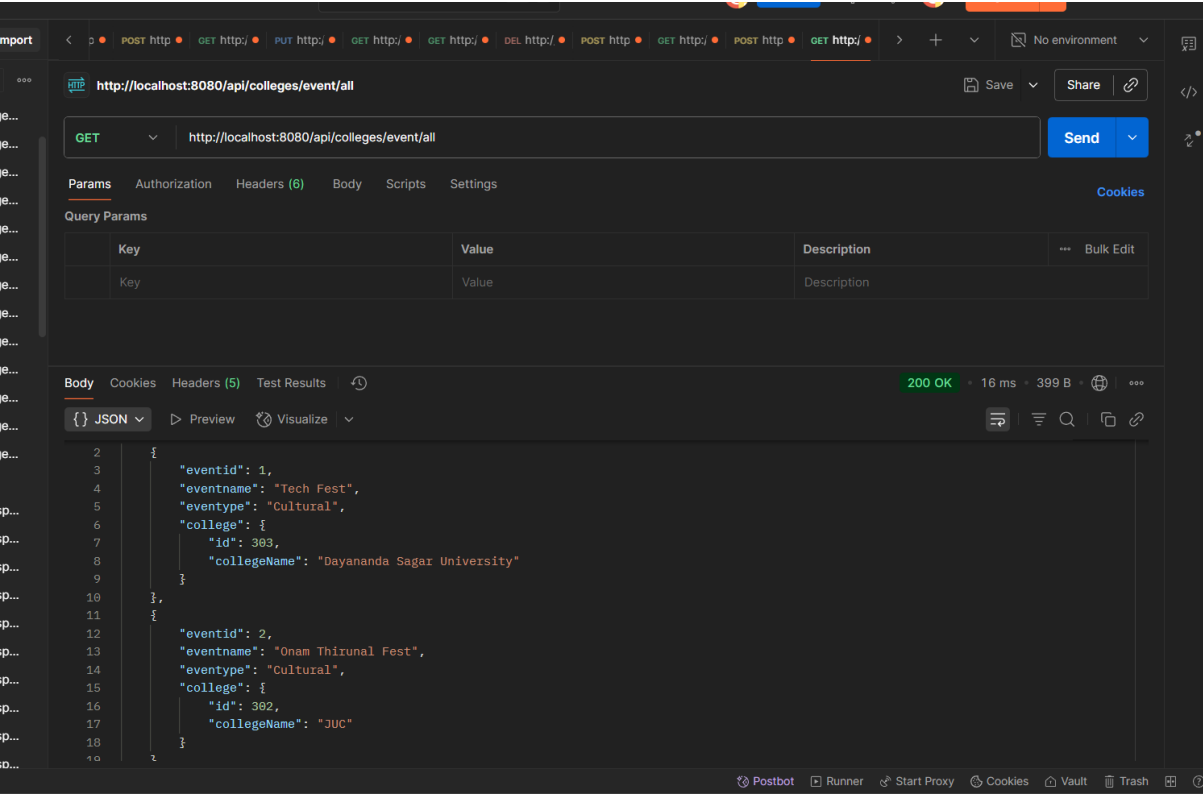
Listing out all colleges registered from the DB using GET



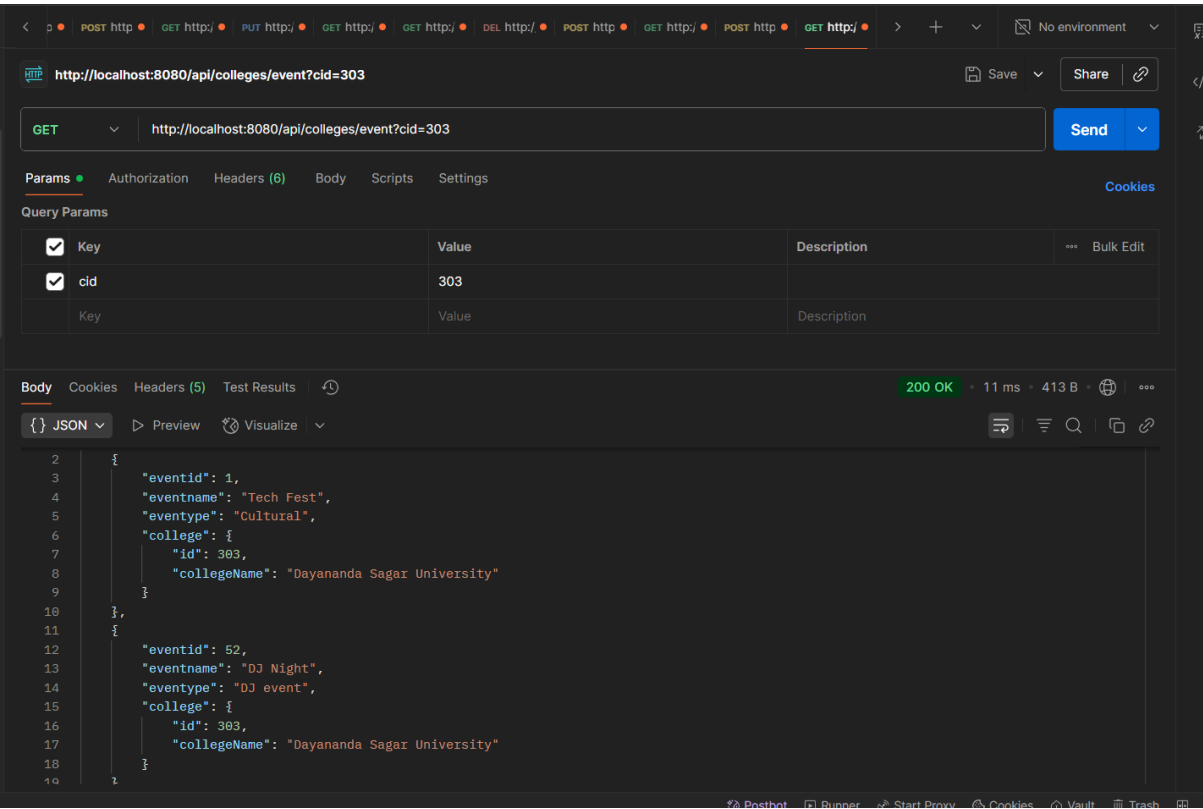
Update colleges



Creating an Event and mapped event entity to college entity by many to one mapping since there can be many events in one college (the model config will be added further)



Listing all the events from all events from all colleges



Finding events from a specific college id using requestparam

The screenshot displays an HTTP client interface with a PUT request to `http://localhost:8080/api/colleges/event/admin/update?EventID=1`. The request body is a JSON object with the following structure:

```
{
  "eventid": 1,
  "eventname": "Tech Fest 2025",
  "eventtype": "Cultural",
  "college": {
    "id": 303
  }
}
```

The response status is **200 OK** with a response time of 331 ms and a body size of 292 B. The response body is a JSON object with the following structure:

```
{
  "eventid": 1,
  "eventname": "Tech Fest 2025",
  "eventtype": "Cultural",
  "college": {
    "id": 303,
    "collegeName": "Dayananda Sagar University"
  }
}
```

Update a events details by PUT using eventid as requestparam

Student And Registration API Endpoints:

The screenshot displays a REST client interface with a POST request to `http://localhost:8080/api/colleges/student/create`. The request body is a JSON object representing a student. The response is a 200 OK status with a JSON object representing the created student, including an assigned ID and the college name.

Request:

```
POST http://localhost:8080/api/colleges/student/create
```

Request Body (JSON):

```
{
  "sname": "Giridharan KS",
  "usn": "ENG22CS0306",
  "college": {
    "id": 303
  }
}
```

Response:

```
200 OK • 559 ms • 280 B • [Status Bar Icons]
```

Response Body (JSON):

```
{
  "id": 1,
  "sname": "Giridharan KS",
  "usn": "ENG22CS0306",
  "college": {
    "id": 303,
    "collegeName": "Dayananda Sagar University"
  }
}
```

Creating student and mapping student entity to college using many to one mapping as there can be many students from one college

The screenshot shows an HTTP client interface with the following details:

- URL:** `http://localhost:8080/api/colleges/student/update?studentID=1`
- Method:** `PUT`
- Body (raw):**

```
1 {
2   "sname": "Giridharan Kandasamy",
3   "usn": "ENG22CS0306",
4   "college": {
5     "id": 303
6   }
7 }
```
- Response:** `200 OK` (36 ms, 287 B)
- Response Body (JSON):**

```
1 {
2   "id": 1,
3   "sname": "Giridharan Kandasamy",
4   "usn": "ENG22CS0306",
5   "college": {
6     "id": 303,
7     "collegeName": "Dayananda Sagar University"
8   }
9 }
```

Updating student info using the student id

HTTP **DELETE** `http://localhost:8080/api/colleges/student/delete?studentID=2` Save Share

Send

Params • Auth Headers (6) Body Scripts Settings Cookies

Query Params

| <input checked="" type="checkbox"/> | Key | Value | Description | ... | Bulk Edit |
|-------------------------------------|-----------|-------|-------------|-----|-----------|
| <input checked="" type="checkbox"/> | studentID | 2 | | | |
| | Key | Value | Description | | |

Body Raw Preview Visualize 200 OK 35 ms 177 B ...

1 deletion done

Deleting student info from the table using a requestparam and DeleteMapping

HTTP **POST** `http://localhost:8080/api/colleges/event/regs/register` Save Share

Send

Params Authorization Headers (6) Body • Scripts Settings Cookies Beautify

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL ☐ JSON

1 {

Body Cookies Headers (5) Test Results 200 OK 436 ms 490 B ...

JSON Preview Visualize

```

1 {
2   "regid": 1,
3   "event": {
4     "eventId": 1,
5     "eventName": "Tech Fest 2025",
6     "eventtype": "Cultural",
7     "college": {
8       "id": 303,
9       "collegeName": "Dayananda Sagar University"
10    }
11  },
12  "student": {
13    "id": 1,
14    "sname": "Giridharan Kandasamy",
15    "usn": "ENG22CS0306",
16    "college": {
17      "id": 303,
18      "collegeName": "Dayananda Sagar University"
19    }
20  },
21  "status": "registered",
22  "attendance": "absent"
23 }
```

Registration added and mapped with event and student by many to one mapping

Attendance report:

The screenshot shows the Postman interface with a PUT request to `http://localhost:8080/api/attendance/mark/1?status=present`. The request is successful, returning a `200 OK` status. The response body is raw text: "Attendance marked as present".

Query Params

| Key | Value | Description |
|--------|---------|-------------|
| status | present | |

Body

```
1 Attendance marked as present
```

Marked attendance using registration id and generated attendance report

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/attendance/summary/1`. The request is successful, returning a `200 OK` status. The response body is JSON, showing attendance counts.

Query Params

| Key | Value | Description |
|-----|-------|-------------|
| | | |

Body

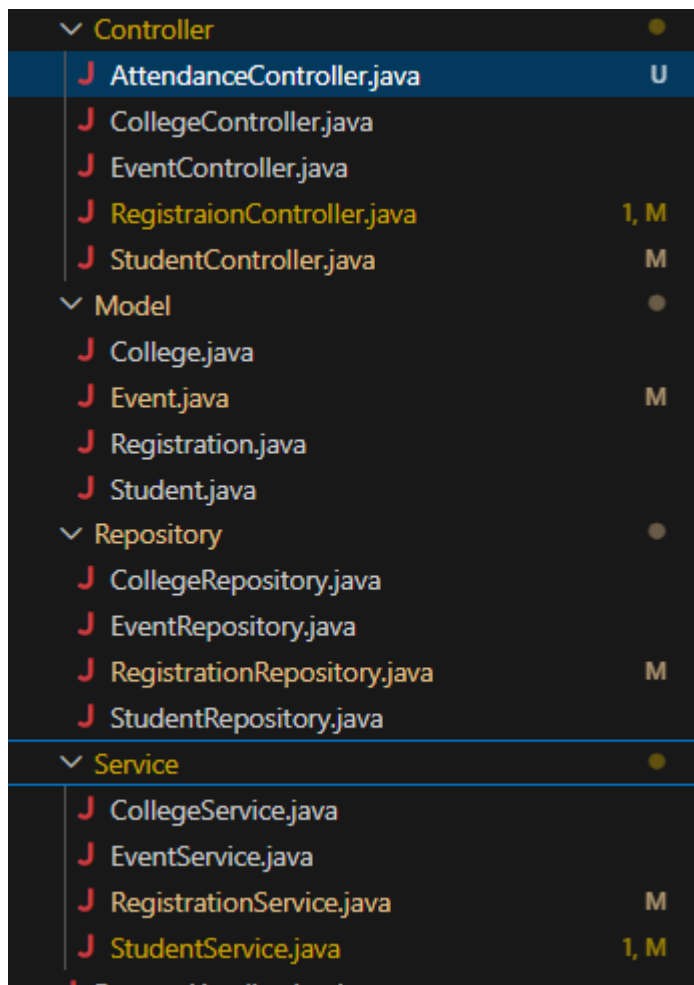
```
1 {
2   "present_count": 1,
3   "total_registered": 1,
4   "absent_count": 0
5 }
```


The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8080/api/attendance/present/1`
- Method:** `GET`
- Status:** `200 OK` (94 ms, 493 B)
- Response Body (JSON):**

```
{  "eventname": "Tech Fest 2025",  "eventtype": "Cultural",  "college": {    "id": 303,    "collegeName": "Dayananda Sagar University"  },  "student": {    "id": 1,    "sname": "Giridharan Kandasamy",    "usn": "ENG22CS0306",    "college": {      "id": 303,      "collegeName": "Dayananda Sagar University"    }  },  "status": "registered",  "attendance": "present"}
```

Getting a list of present and absent students using attendance api endpoints



File Structure

Models(Entity):

College:

```
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class College {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column(unique = true, nullable = false)
    public String collegeName;
}
```

Event:

```
@Entity
@Data
public class Event {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long eventid;
    @NotBlank(message = "event name should be entered")
    String eventname;
    @NotBlank(message = "event type should be selected")
    String eventtype;
    @ManyToOne
    @JoinColumn(name = "collegeID")
    private College college;
}
```

Registration:

```
@Entity
@Data
public class Registration {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long regid;

    @ManyToOne
    @JoinColumn(name = "eventID")
    Event event;

    @ManyToOne
    @JoinColumn(name = "StudentID")
    Student student;

    @Column(nullable = false)
    String status = "registered";

    @Column(nullable = false)
    String attendance = "absent";
}
```

Student:

```
@Entity
@Data
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column(unique = true, nullable = false)
    public String sname;

    @Column(unique = true, nullable = false)
    public String usn;

    @ManyToOne
    @JoinColumn(name = "collegeID")
    private College college;
}
```

Controllers:

College:

```
@RestController
@RequestMapping("/api/colleges")
public class CollegeController {

    @Autowired
    CollegeService collegeService;

    @PostMapping("/add")
    public ResponseEntity<> addCollege(@RequestBody College c){
        ResponseEntity<College> res = new ResponseEntity<>(collegeService.addCollege(c),HttpStatus.CREATED);

        if(!res.hasBody()){
            return new ResponseEntity<>("college details not added try again",HttpStatus.CONFLICT);
        }

        return res;
    }

    @GetMapping("/all")
    public ResponseEntity<List<College>> getAll(){
        return new ResponseEntity<>(collegeService.getColleges(),HttpStatus.OK);
    }

    @DeleteMapping("/delete")
    public ResponseEntity<String> deleteCollege(@RequestParam("collegeid") long id ){
        try{
            collegeService.deleteCollege(id);
            return new ResponseEntity<>("deletion done!",HttpStatus.OK);
        } catch (Exception e){
            return new ResponseEntity<>("deletion failed",HttpStatus.CONFLICT);
        }
    }

    @PutMapping("/update")
    public ResponseEntity<> updateCollege(@RequestParam("collegeID") Long id , @RequestBody College college){
        ResponseEntity<College> res = new ResponseEntity<>(collegeService.updateCollege(id, college),HttpStatus.OK);

        if(res.getBody() == null){
            return new ResponseEntity<>("not found",HttpStatus.NOT_FOUND);
        }

        return res;
    }
}
```

Event:

```

@RestController
@RequestMapping("api/colleges/event")
public class EventController {

    @Autowired
    private EventService eventService;
    @Autowired
    private CollegeRepository collegeDB;
    @PostMapping("/admin/create")
    public ResponseEntity<> createEvent(@RequestBody Event event){
        College college = collegeDB.findById(event.getCollege().getId()).orElse(other:null);
        if(college == null){
            return new ResponseEntity<>("no college found!",HttpStatus.NOT_FOUND);
        }
        event.setCollege(college);
        return new ResponseEntity<>(eventService.createEvent(event),HttpStatus.CREATED);
    }
    @GetMapping("/all")
    public ResponseEntity<List<Event>> getAll(){
        return new ResponseEntity<>(eventService.allEvents(),HttpStatus.OK);
    }
    @GetMapping("")
    public ResponseEntity<List<Event>> getAllEvents(@RequestParam("cid") Long id){
        return new ResponseEntity<>(eventService.getAllEventsbbyid(id),HttpStatus.OK);
    }
    @PutMapping("/admin/update")
    public ResponseEntity<> updateEvent(@RequestParam("EventID") long eid, @RequestBody Event e){
        ResponseEntity<Event> res = new ResponseEntity<>(eventService.update(eid, e),HttpStatus.OK);
        if(res.getBody() == null){
            return new ResponseEntity<>("not found",HttpStatus.NOT_FOUND);
        }
        return res;
    }

    @DeleteMapping("/admin/delete")
    public ResponseEntity<String> deleteEvent(@RequestParam("eventID") long id){
        try{
            eventService.deleteEvent(id);
            return new ResponseEntity<>("deletion done",HttpStatus.OK);
        } catch (Exception e){
            return new ResponseEntity<>("deletion failed",HttpStatus.NOT_FOUND);
        }
    }
}

```

Registration:

```

@RestController
@RequestMapping("api/colleges/event/regist")

public class RegistraionController {
    @Autowired
    private EventRepository EventDB;
    @Autowired
    private StudentRepository StudentDB;
    @Autowired
    private RegistrationService RegService;

    @PostMapping("/register")
    public ResponseEntity<?> createRegis(@RequestBody Registration reg){
        if (reg.getStudent().getId() == null) {
            return ResponseEntity.badRequest().body("Student id must not be null");
        }
        if (reg.getEvent().getEventid() == null) {
            return ResponseEntity.badRequest().body("Event id must not be null");
        }
        Student student = StudentDB.findById(reg.getStudent().getId()).orElse(other:null);
        Event event = EventDB.findById(reg.getEvent().getEventid()).orElse(other:null);
        if(event == null || student == null){
            return new ResponseEntity<>("not found check with the student id or event id",HttpStatus.NOT_FOUND);
        }
        reg.setStudent(student);
        reg.setEvent(event);
        return new ResponseEntity<>(RegService.createRegis(reg),HttpStatus.OK);
    }

    @PutMapping("/cancel")
    public ResponseEntity<?> CancelRegistration(@RequestParam("regID") long id){
        Registration reg = RegService.getByRegid(id);
        if(reg == null){
            return new ResponseEntity<>("not found",HttpStatus.NOT_FOUND);
        }
        reg.setStatus(status:"cancelled");
        return new ResponseEntity<>(RegService.updateStatus(reg),HttpStatus.OK);
    }

    @GetMapping("/student")
    public ResponseEntity<List<Registration>> getStudentRegis(@RequestParam("studentID") long id){
        return new ResponseEntity<>(RegService.allRegis(id),HttpStatus.OK);
    }
}

```

Student:

```
eventapi / src / main / java / com / events / eventapi / Controller / StudentController.java / StudentController / GetInfo(long)
20 @RestController
21 @RequestMapping("api/colleges/student")
22 public class StudentController {
23
24     @Autowired
25     StudentService studentService;
26     @Autowired
27     private CollegeRepository collegeDB;
28     @PostMapping("/create")
29     public ResponseEntity<> createStudent(@RequestBody Student student){
30         College college = collegeDB.findById(student.getCollege().getId()).orElse(other:null);
31         if(college == null){
32             return new ResponseEntity<>("college id is invalid or not found",HttpStatus.NOT_FOUND);
33         }
34         student.setCollege(college);
35         return new ResponseEntity<>(studentService.createStudent(student),HttpStatus.OK);
36     }
37     @GetMapping("/info")
38     public ResponseEntity<> GetInfo(@RequestParam("studentID") long id){
39         ResponseEntity<Student> res = new ResponseEntity<>(studentService.getStudentbyID(id),HttpStatus.OK);
40         if(res.getBody() == null){
41             return new ResponseEntity<>("student info not found",HttpStatus.NOT_FOUND);
42         }
43         return res;
44     }
45 }
46 @PutMapping("/update")
47 public ResponseEntity<> UpdateInfo(@RequestParam("studentID") long id, @RequestBody Student s){
48     ResponseEntity<Student> res = new ResponseEntity<>(studentService.updateInfo(id,s),HttpStatus.OK);
49     if(res.getBody() == null){
50         return new ResponseEntity<>("not found",HttpStatus.NOT_FOUND);
51     }
52     return res;
53 }
54
55 @DeleteMapping("/delete")
56 public ResponseEntity<String> DeleteStudent(@RequestParam("studentID") long id){
57     try{
58         studentService.deleteStudent(id);
59         return new ResponseEntity<>("deletion done",HttpStatus.OK);
60     } catch(Exception e){
61         return new ResponseEntity<>("deltion failed",HttpStatus.NOT_FOUND);
62     }
63 }
64 }
```

Attendance:

```

14 @RestController
15 @RequestMapping("/api/attendance")
16 public class AttendanceController {
17     @Autowired
18     private RegistrationRepository registrationRepo;
19     @PutMapping("/mark/{regId}/")
20     public ResponseEntity<> markAttendance(@PathVariable Long regId,
21                                           @RequestParam String status) {
22         Registration reg = registrationRepo.findById(regId).orElse(other:null);
23         if (reg == null) {
24             return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Registration not found");
25         }
26
27         if (!status.equalsIgnoreCase(anotherString:"present") && !status.equalsIgnoreCase(anotherString:"absent")) {
28             return ResponseEntity.badRequest().body("Status must be 'present' or 'absent'");
29         }
30
31         reg.setAttendance(status.toLowerCase());
32         registrationRepo.save(reg);
33
34         return ResponseEntity.ok("Attendance marked as " + status);
35     }
36
37     @GetMapping("/summary/{eventId}")
38     public ResponseEntity<> getEventSummary(@PathVariable Long eventId) {
39         long total = registrationRepo.countByEventEventid(eventId);
40         long present = registrationRepo.countByEventEventidAndAttendance(eventId, attendance:"present");
41         long absent = registrationRepo.countByEventEventidAndAttendance(eventId, attendance:"absent");
42
43         Map<String, Object> summary = new HashMap<>();
44         summary.put(key:"total_registered", total);
45         summary.put(key:"present_count", present);
46         summary.put(key:"absent_count", absent);
47
48         return ResponseEntity.ok(summary);
49     }
50
51     @GetMapping("/present/{eventId}")
52     public ResponseEntity<> getPresentList(@PathVariable Long eventId) {
53         return ResponseEntity.ok(registrationRepo.findByEventEventidAndAttendance(eventId, attendance:"present"));
54     }
55
56     @GetMapping("/absent/{eventId}")
57     public ResponseEntity<> getAbsentList(@PathVariable Long eventId) {
58         return ResponseEntity.ok(registrationRepo.findByEventEventidAndAttendance(eventId, attendance:"absent"));
59     }
60 }

```


Repositories:

College:

```
import com.events.Eventcap11.model.College;

@Repository
public interface CollegeRepository extends JpaRepository<College,Long> {

}
```

Event:

```
public interface EventRepository extends JpaRepository<Event,Long> {
    List<Event> findByCollegeId(Long CollegeId);
}
```

Registration:

```
public interface RegistrationRepository extends JpaRepository<Registration,Long>{
    List<Registration> findByStudentId(Long id);
    List<Registration> findByEventEventid(Long eventid);
    long countByEventEventid(Long eventid);
    long countByEventEventidAndAttendance(Long eventid, String attendance);
    List<Registration> findByEventEventidAndAttendance(Long eventid, String attendance);
}
```

Student:

```
public interface StudentRepository extends JpaRepository<Student,Long>{

}
```

Service:

College:

```
@Service
public class CollegeService {

    @Autowired
    CollegeRepository collegeDB;

    public College addCollege(College college){
        return collegeDB.save(college);
    }

    public List<College> getColleges(){
        return collegeDB.findAll();
    }

    public College getbyID(Long id){
        return collegeDB.findById(id).orElse(other:null);
    }

    public void deletecollege(long id) throws Exception{
        if(getbyID(id)==null){
            throw new Exception(message:"not found");
        }
        collegeDB.delete(getbyID(id));
    }

    public College updateCollege(long id, College c){
        if(getbyID(id)==null){
            return null;
        }
        c.setId(id);
        return collegeDB.save(c);
    }

}
```

Event:

```
@Service
public class EventService {

    @Autowired
    EventRepository EventDB;

    public Event createEvent(Event e){
        return EventDB.save(e);
    }

    public List<Event> allEvents(){
        return EventDB.findAll();
    }

    public List<Event> getAllEventsbyid(long id){
        return EventDB.findById(id);
    }

    public Event getEventById(long id){
        return EventDB.findById(id).orElse(null);
    }

    public Event update(long id,Event e){
        if(getEventById(id)==null){
            return null;
        }
        e.setEventid(id);
        return EventDB.save(e);
    }

    public void deleteEvent(Long id) throws Exception{
        Event e = getEventById(id);
        if(e == null){
            throw new Exception(message:"no event registered recheck the event id please");
        }
        EventDB.delete(e);
    }
}
```

Registration:

```
@Service
public class RegistrationService {
    @Autowired
    RegistrationRepository RegisDB;

    public Registration createRegis(Registration reg){
        return RegisDB.save(reg);
    }

    public Registration getByRegid(long id){
        return RegisDB.findById(id).orElse(other:null);
    }

    public Registration updateStatus(Registration reg){
        return RegisDB.save(reg);
    }

    public List<Registration> allRegis(long id){
        return RegisDB.findByStudentId(id);
    }
}
```

–Student:

```
@Service
public class StudentService {

    @Autowired
    StudentRepository StudentDB;

    public Student createStudent(Student s){
        return StudentDB.save(s);
    }

    public Student getStudentbyID(Long id){
        return StudentDB.findById(id).orElse(other:null);
    }

    public Student updateInfo(Long id, Student s){
        Student student = getStudentbyID(id);
        if(student == null){
            return null;
        }
        s.setId(id);
        return StudentDB.save(s);
    }

    public void deleteStudent(long id) throws Exception{
        Student student = getStudentbyID(id);
        if(student == null){
            throw new Exception(message:"student not found");
        }
        StudentDB.delete(student);
    }
}
```

| | |
|----------------------------|------|
| Controller | |
| AttendanceController.java | U |
| CollegeController.java | |
| EventController.java | |
| RegistraionController.java | 1, M |
| StudentController.java | M |

For controllers do check:

https://github.com/giridharanks2004/EventRegistration_API_WebKnot

```

7 public class EventapiApplication {
8
9     Run | Debug
10    public static void main(String[] args) {
11        SpringApplication.run(EventapiApplication.class, args);
12    }
13 }
14

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown

```

2025-09-07T13:31:26.681+05:30 INFO 4240 --- [eventapi] [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.t
2025-09-07T13:31:27.023+05:30 INFO 4240 --- [eventapi] [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence u
2025-09-07T13:31:27.701+05:30 WARN 4240 --- [eventapi] [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefo
y be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2025-09-07T13:31:28.266+05:30 INFO 4240 --- [eventapi] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '
2025-09-07T13:31:28.279+05:30 INFO 4240 --- [eventapi] [main] c.events.eventapi.EventapiApplication : Started EventapiApplication in 8.021 seconds (process
PS D:\code\EventRegistration_API_MebKnot> git add .
PS D:\code\EventRegistration_API_MebKnot> git commit -m "update: done with end points and added Attendance controller to generate attendance report"
[main 27747e0] update: done with end points and added Attendance controller to generate attendance report
7 files changed, 155 insertions(+), 4 deletions(-)
create mode 100644 eventapi/src/main/java/com/events/eventapi/Controller/AttendanceController.java
PS D:\code\EventRegistration_API_MebKnot> git push origin main
Enumerating objects: 36, done.
Counting objects: 100% (36/36), done.
Delta compression using up to 12 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (20/20), 3.71 KiB | 543.00 KiB/s, done.
Total 20 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (0/0), completed with 7 local objects.
To https://github.com/giridharanks2804/EventRegistration_API_MebKnot.git
092033e..27747e0 main -> main
PS D:\code\EventRegistration_API_MebKnot>

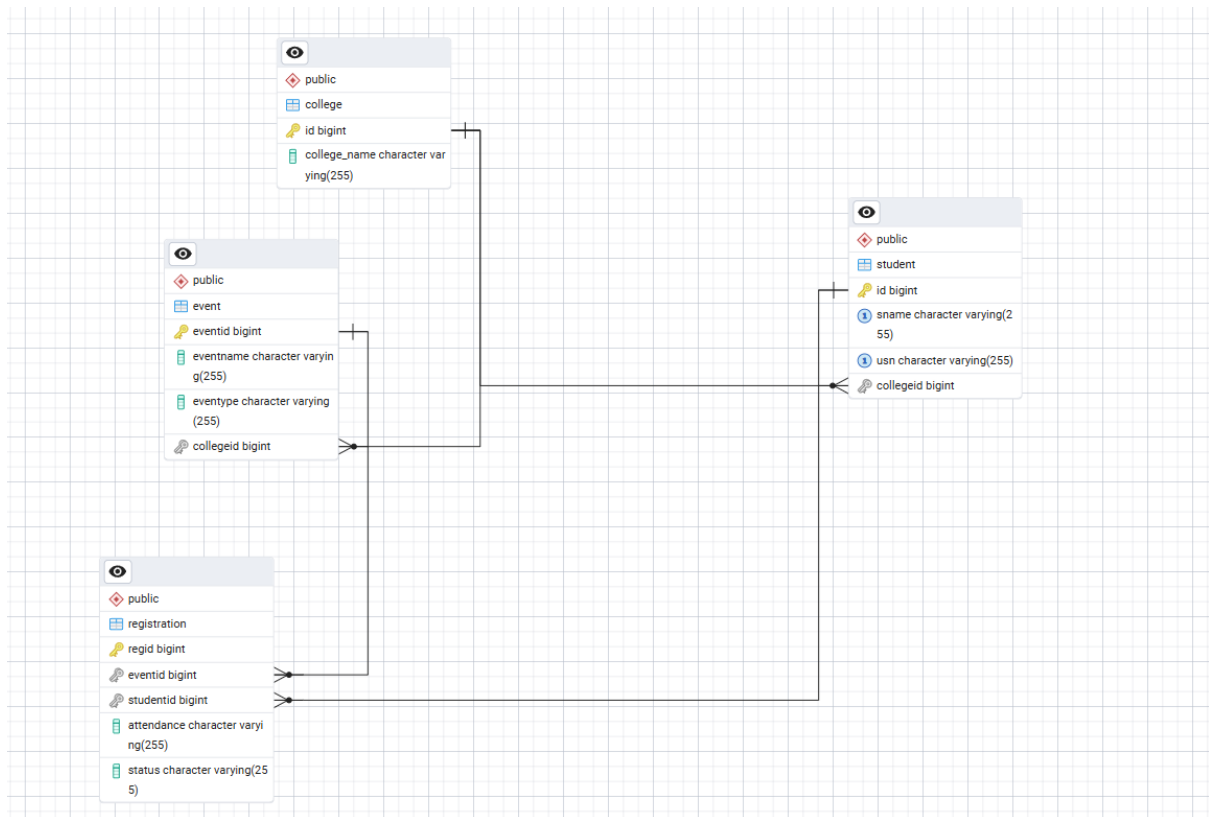
```

Ln 1, Col 1 Tab Size: 4 UTF-8 LF () Java

ENG IN 07-05

Sep 7, 2025, 1:35 PM

Final push before adding documentations



ER Diagram Postgresql