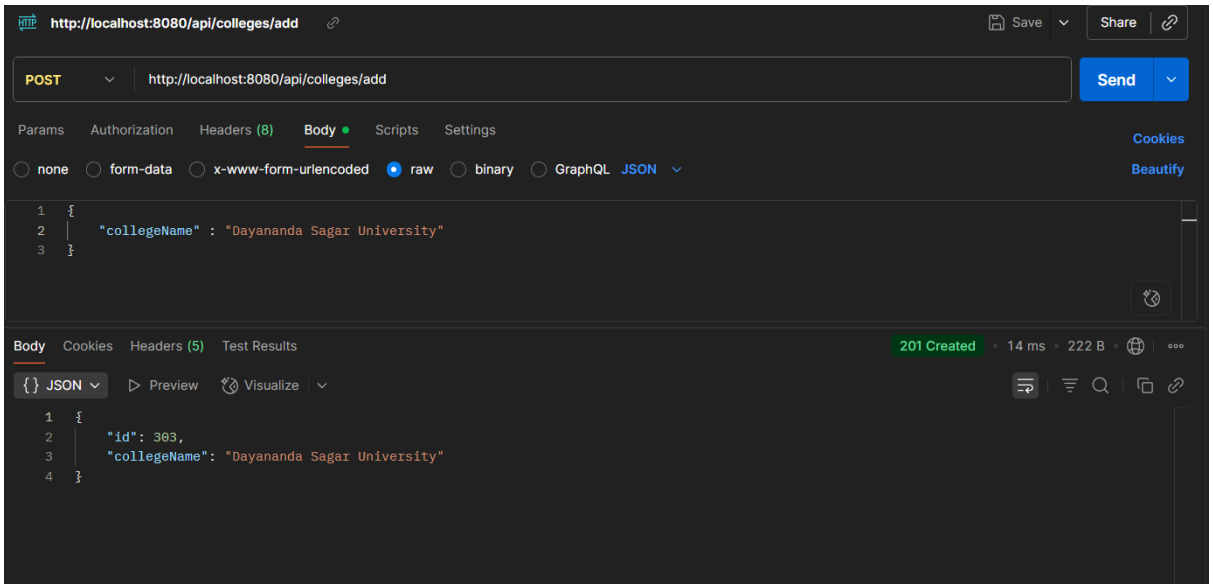
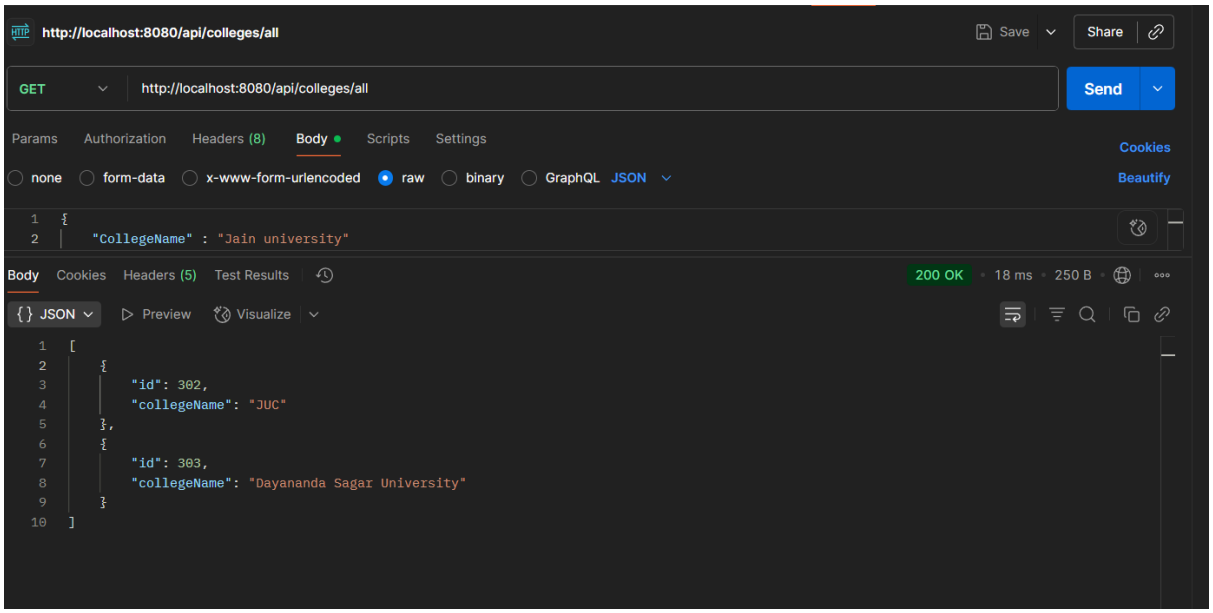
Giridharan KS ENG22CS0306

College And Event API Endpoints:

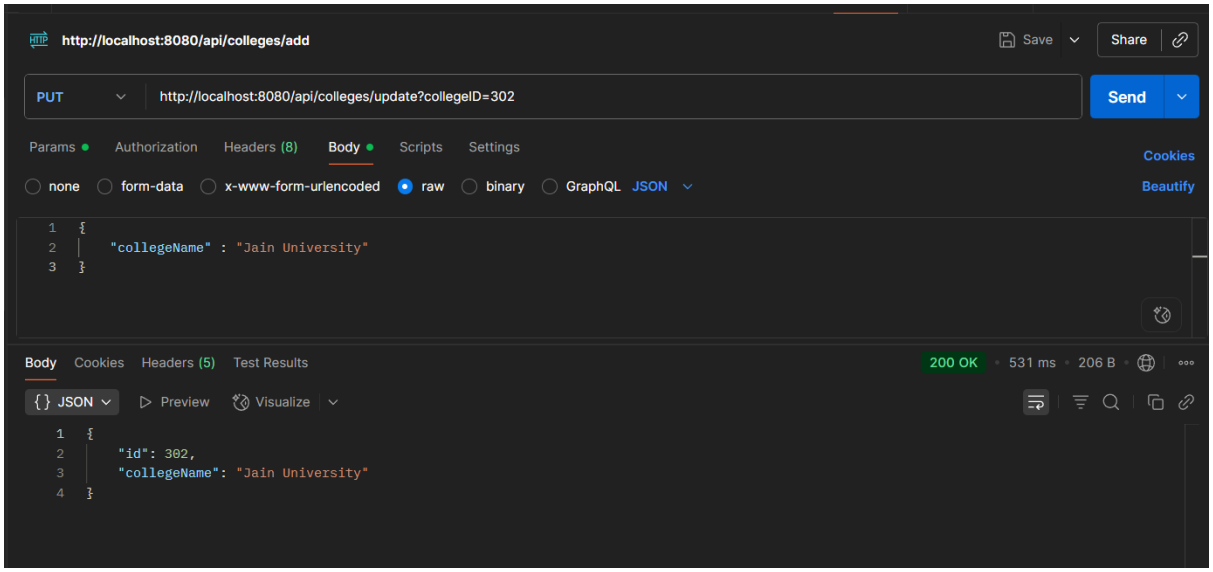


POST method to add colleges with an unique college id

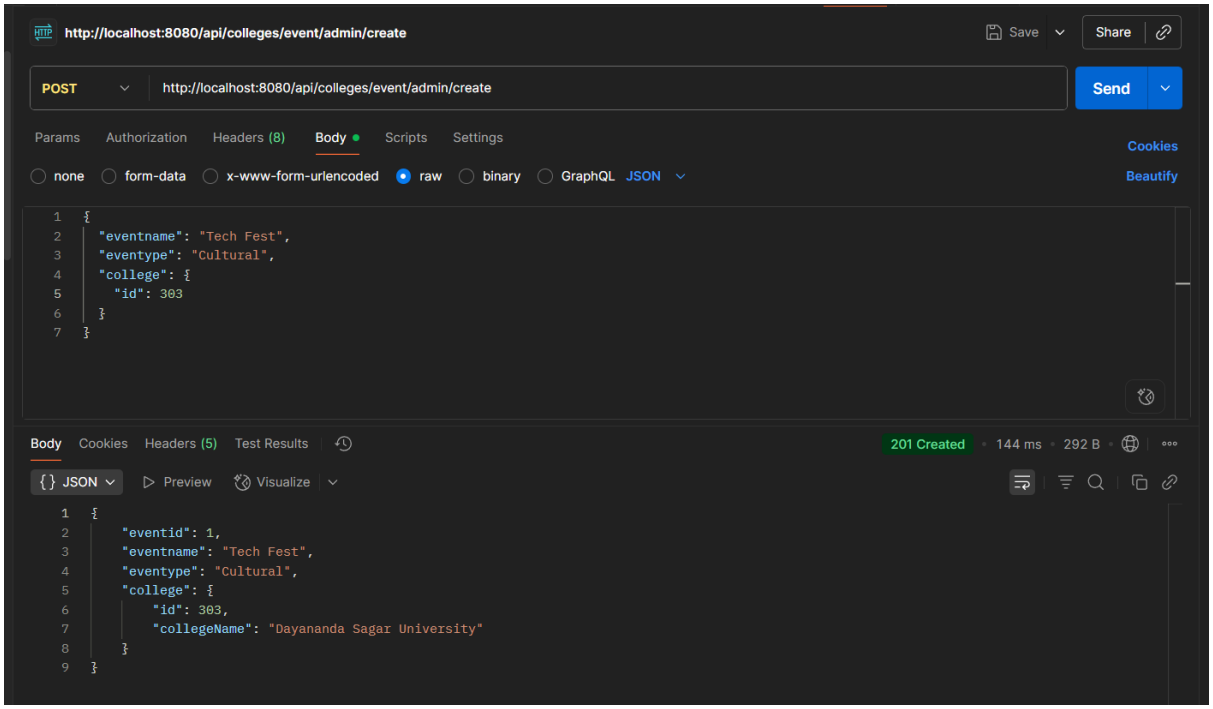


Listing out all colleges registered from the DB using GET

Giridharan KS ENG22CS0306

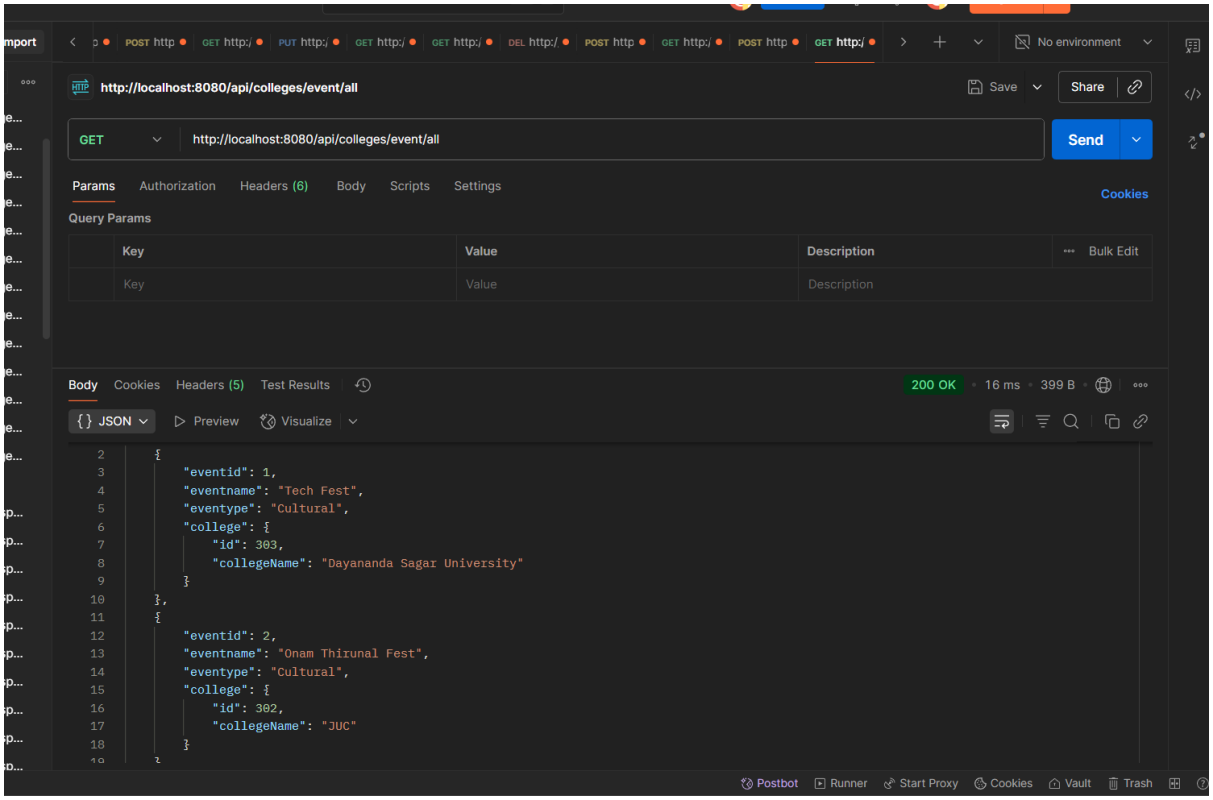


Update colleges

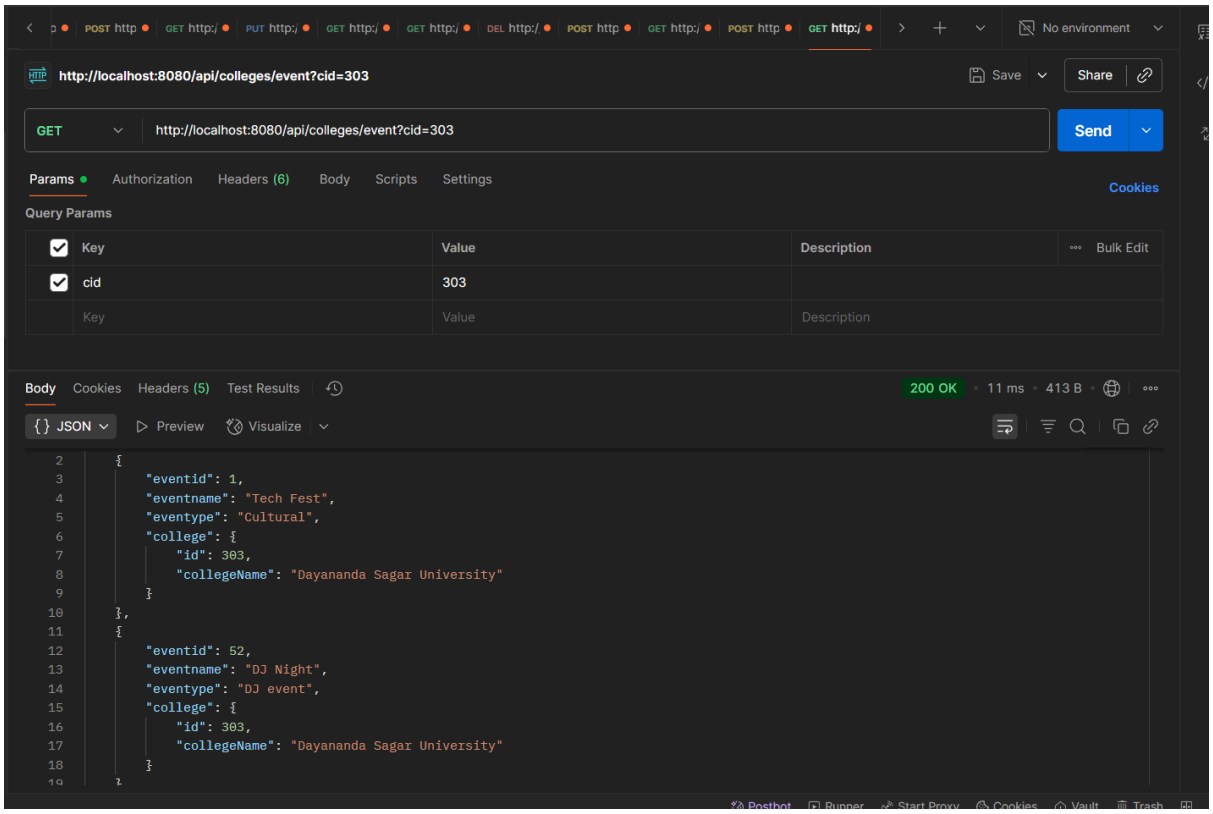


Creating an Event and mapped event entity to college entity by many to one mapping since there can be many events in one college (the model config will be added further)

Giridharan KS ENG22CS0306

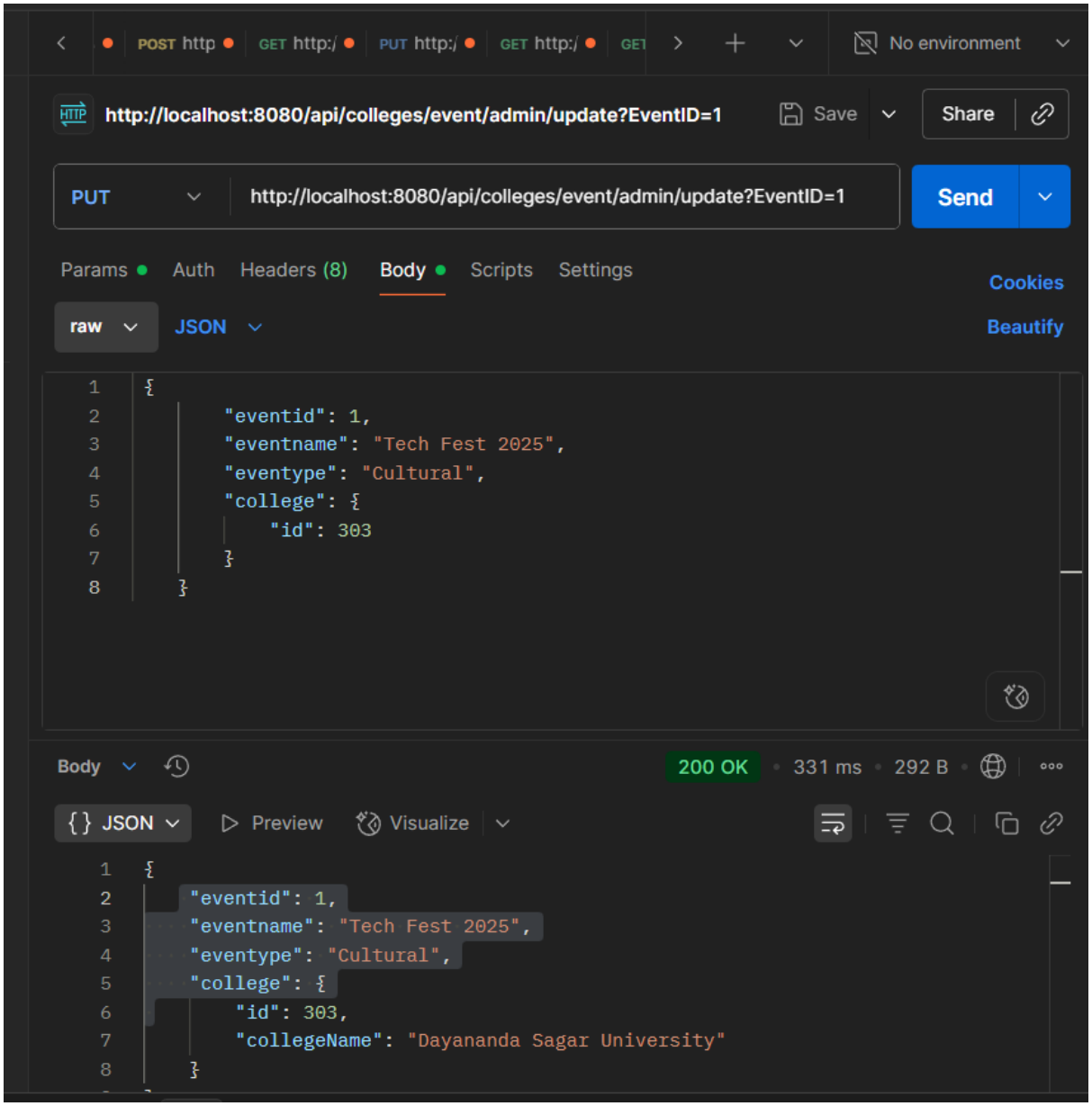


Listing all the events from all events from all colleges



Finding events from a specific college id using requestparam

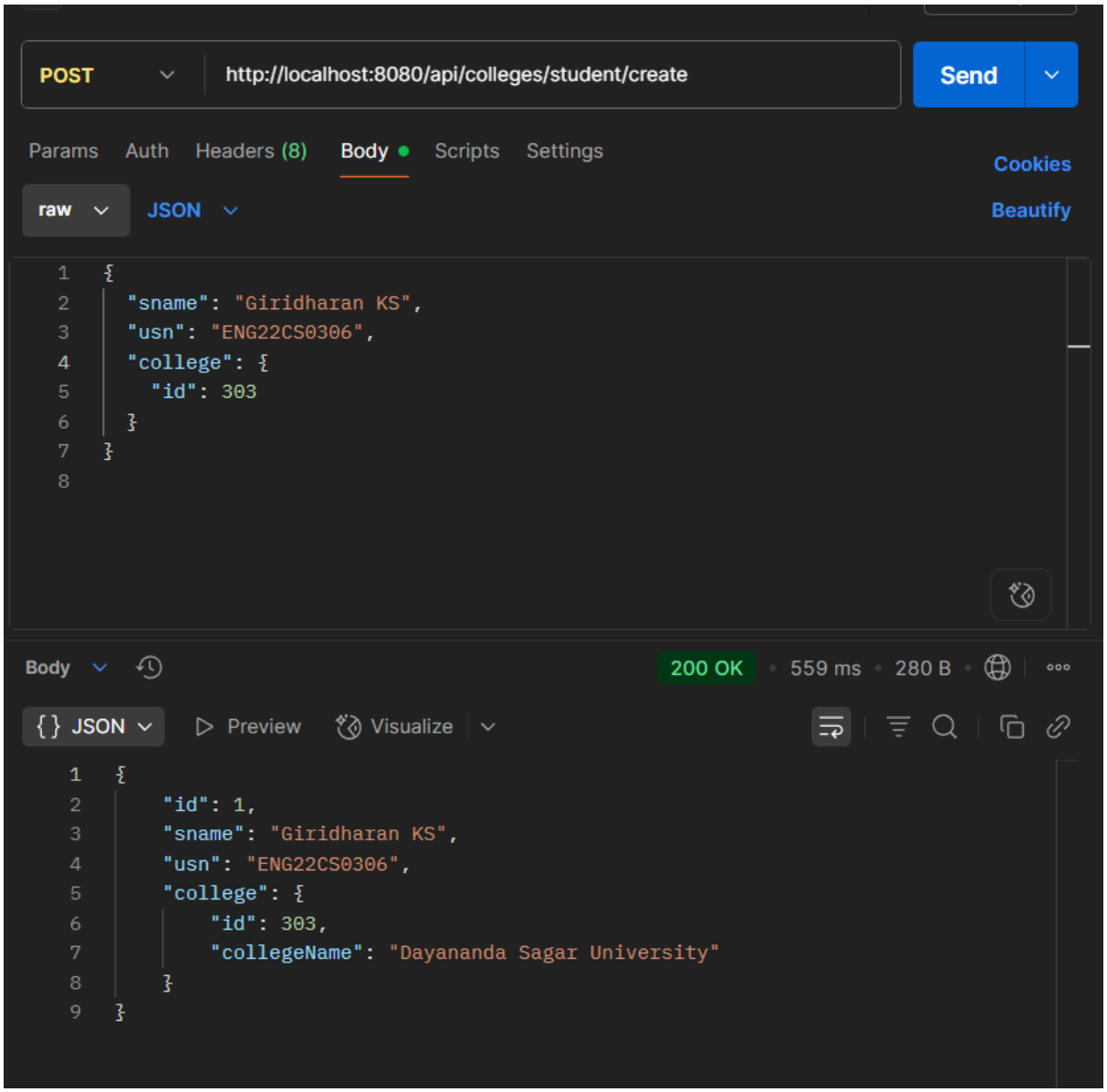
Giridharan KS ENG22CS0306



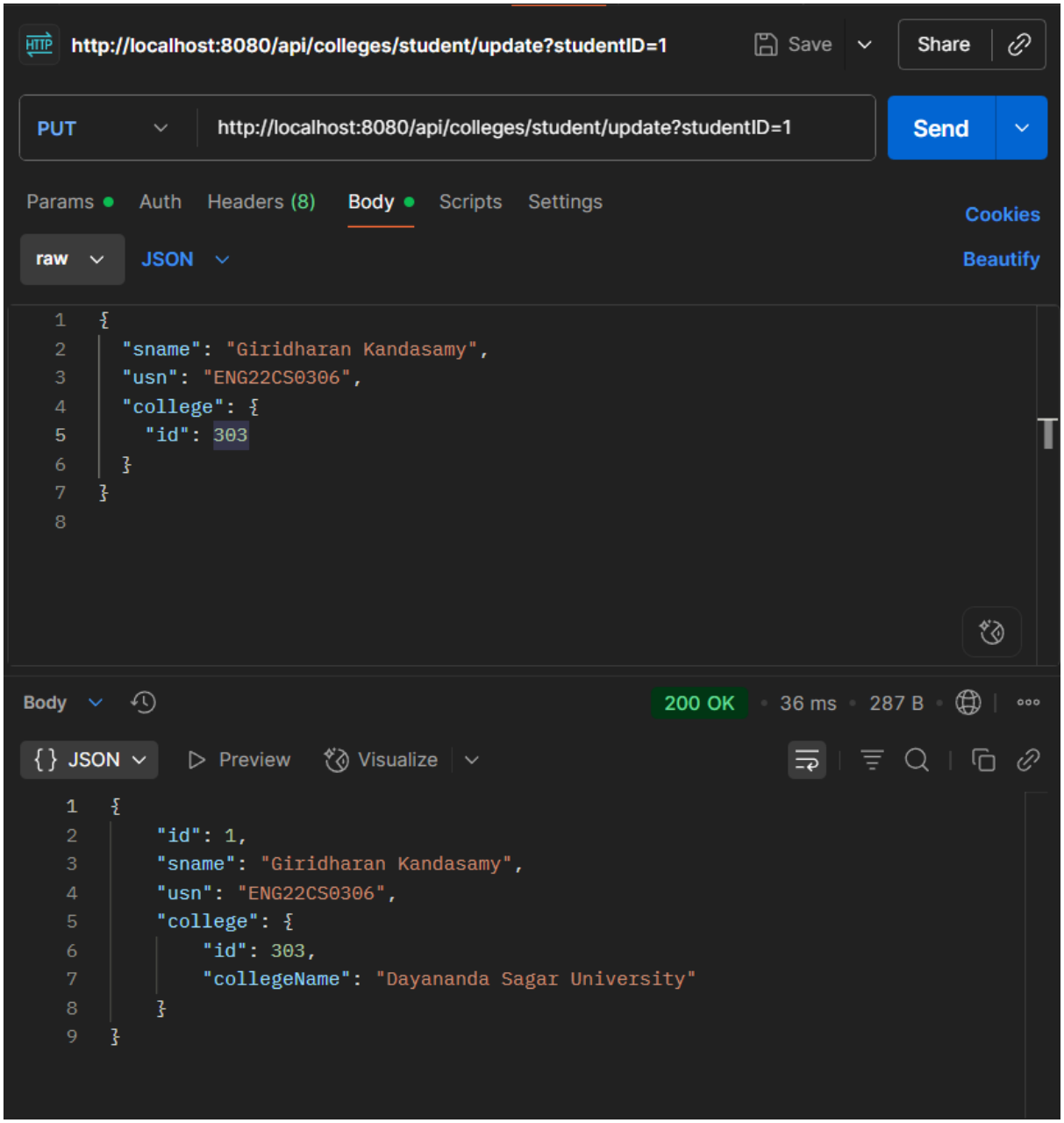
Update a events details by PUT using eventid as requestparam

Giridharan KS ENG22CS0306

Student And Registration API Endpoints:

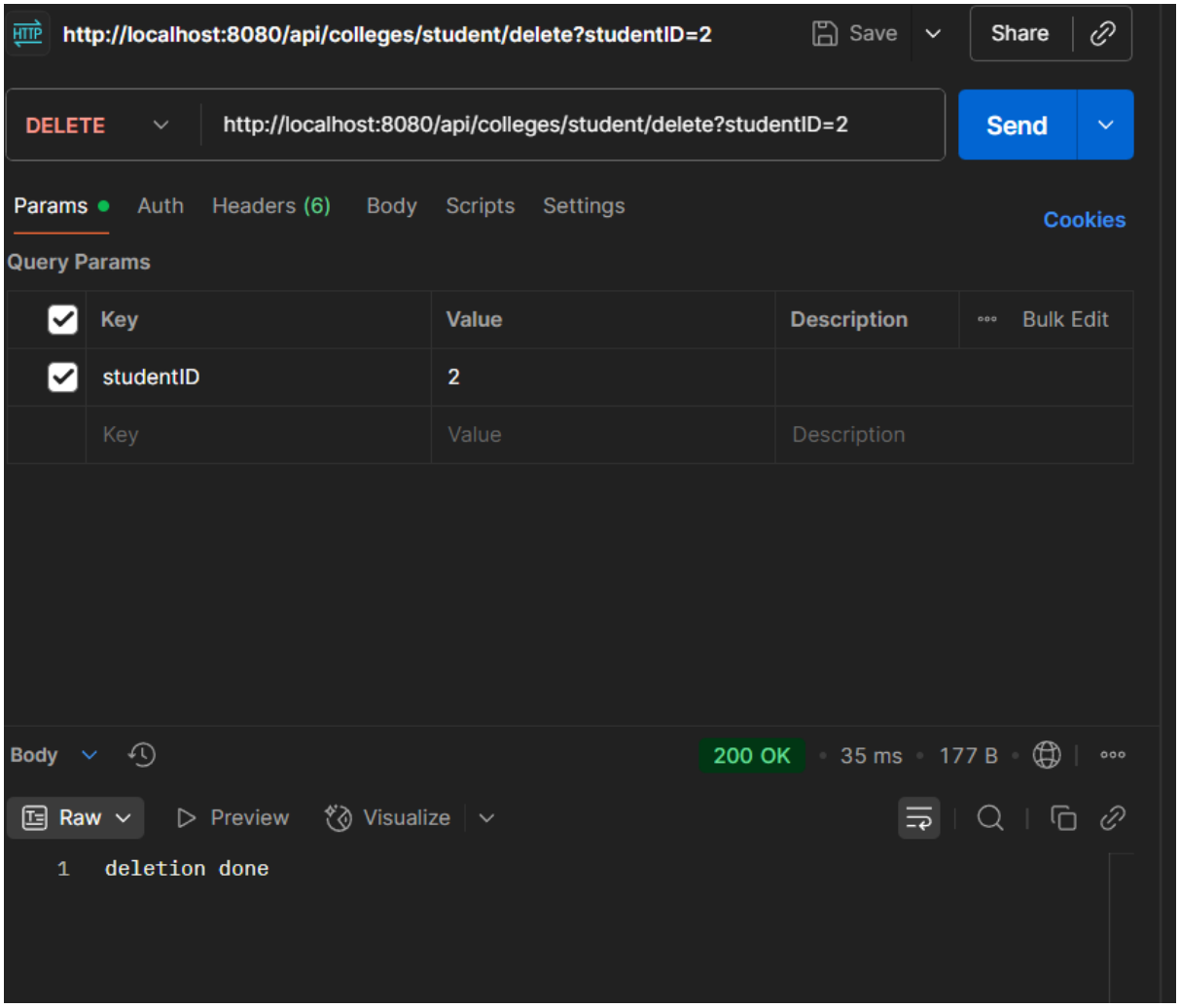


Creating student and mapping student entity to college using many to one mapping as there can be many students from one college

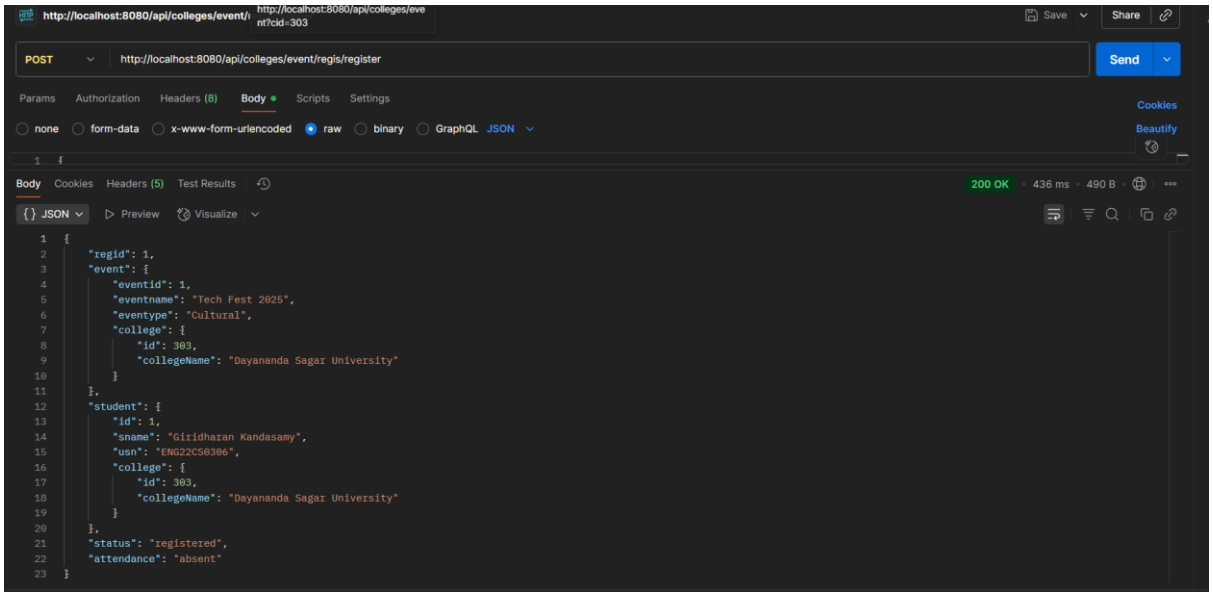


Updating student info using the student id

Giridharan KS ENG22CS0306



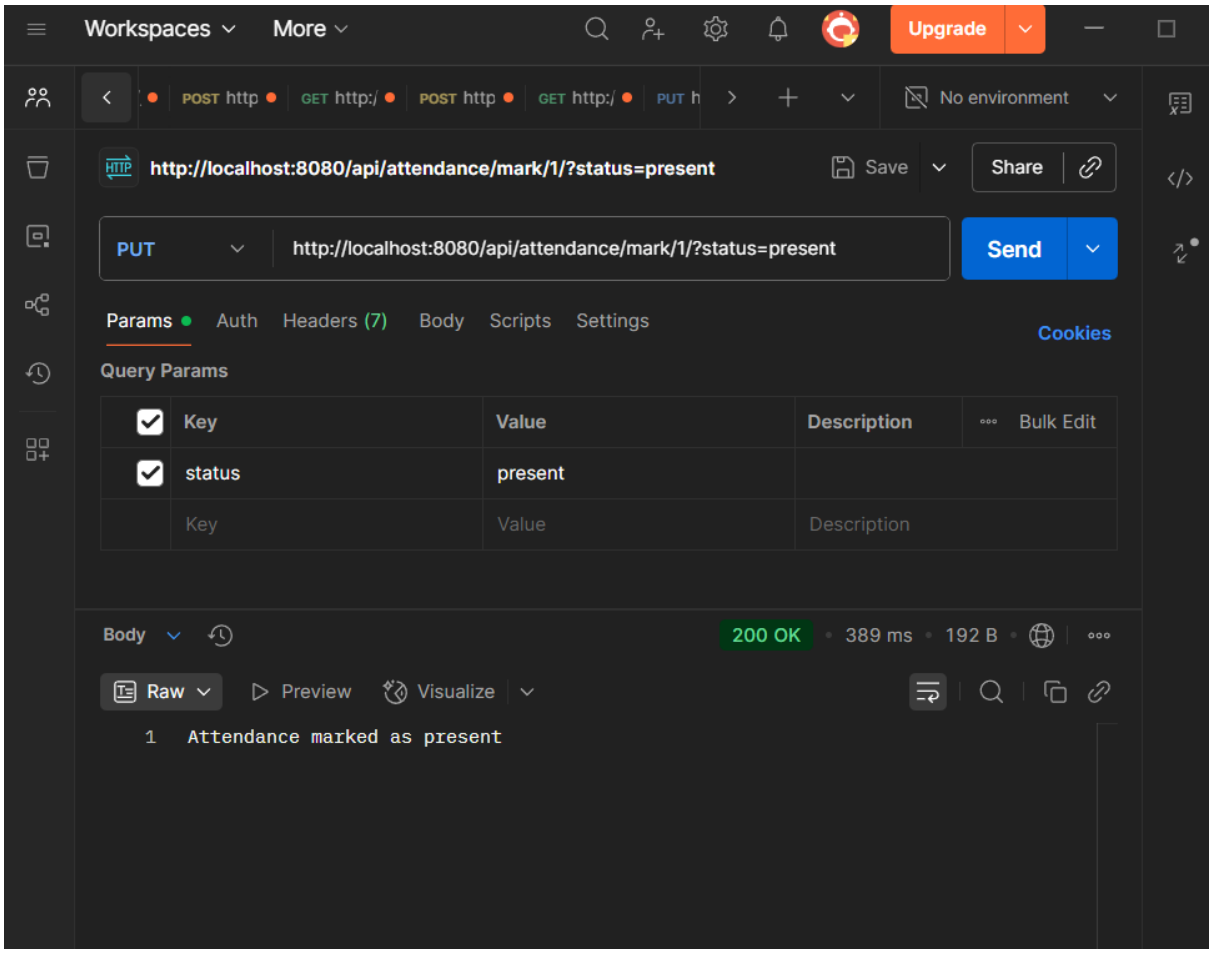
Deleting student info from the table using a requestparam and DeleteMapping



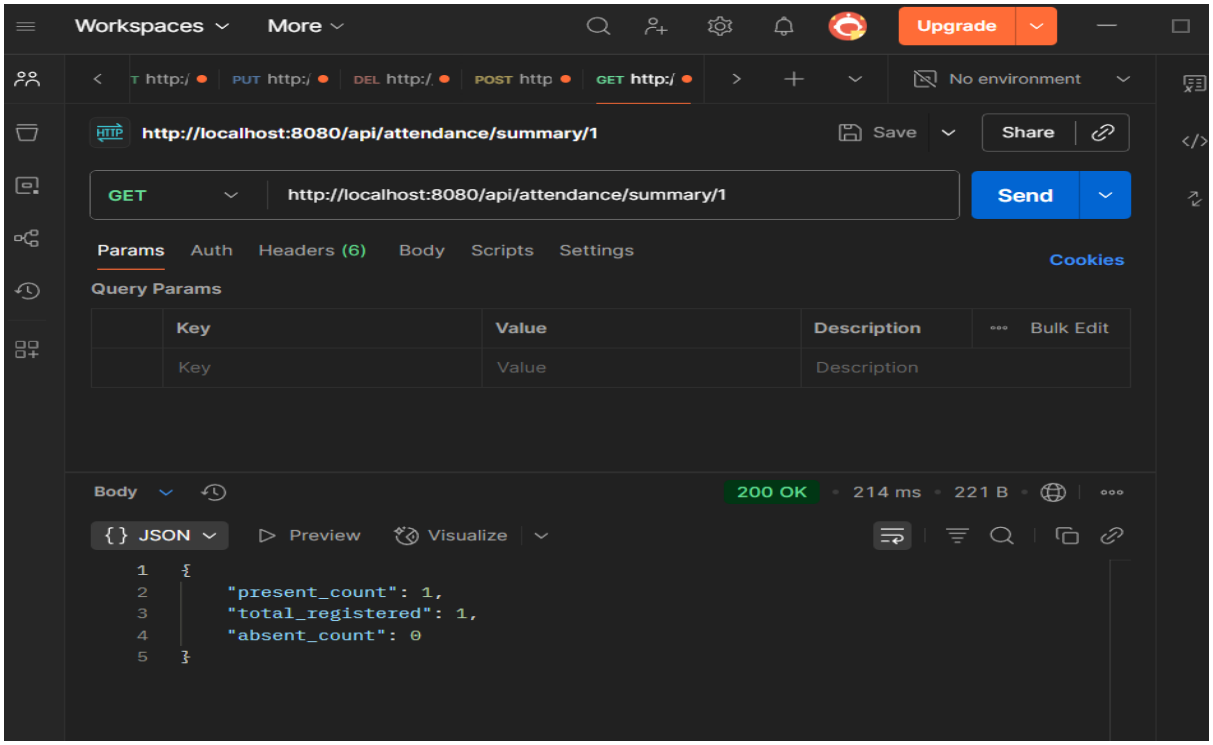
Registration added and mapped with event and student by many to one mapping

Giridharan KS ENG22CS0306

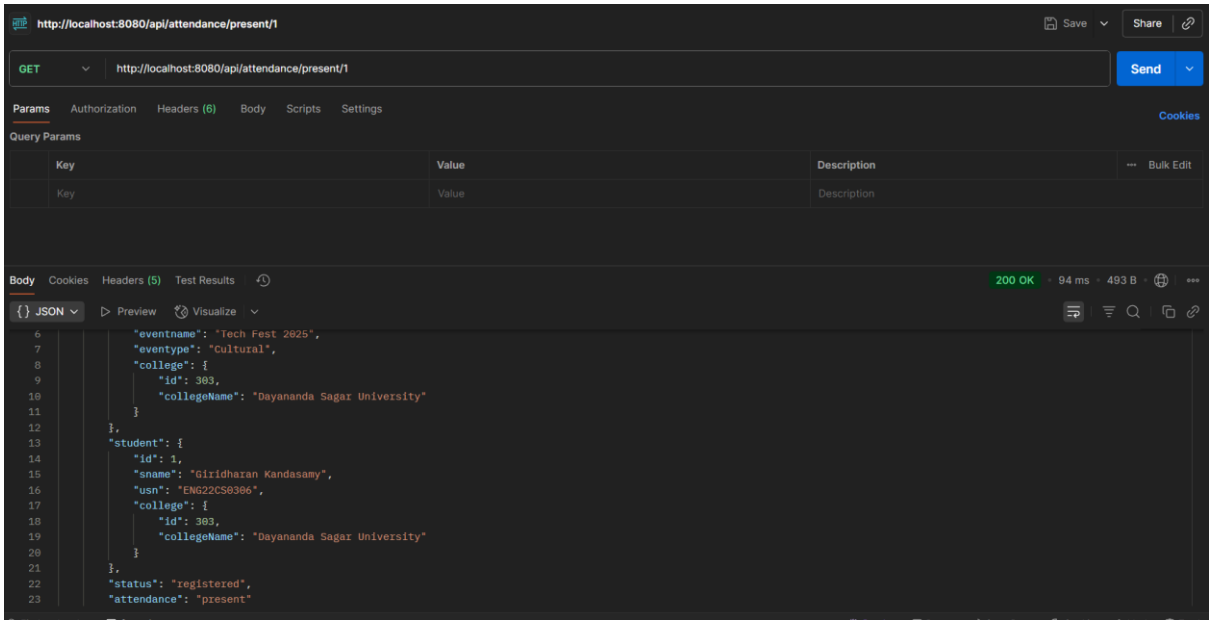
Attendance report:



Marked attendance using regsistration id and generated attendance report

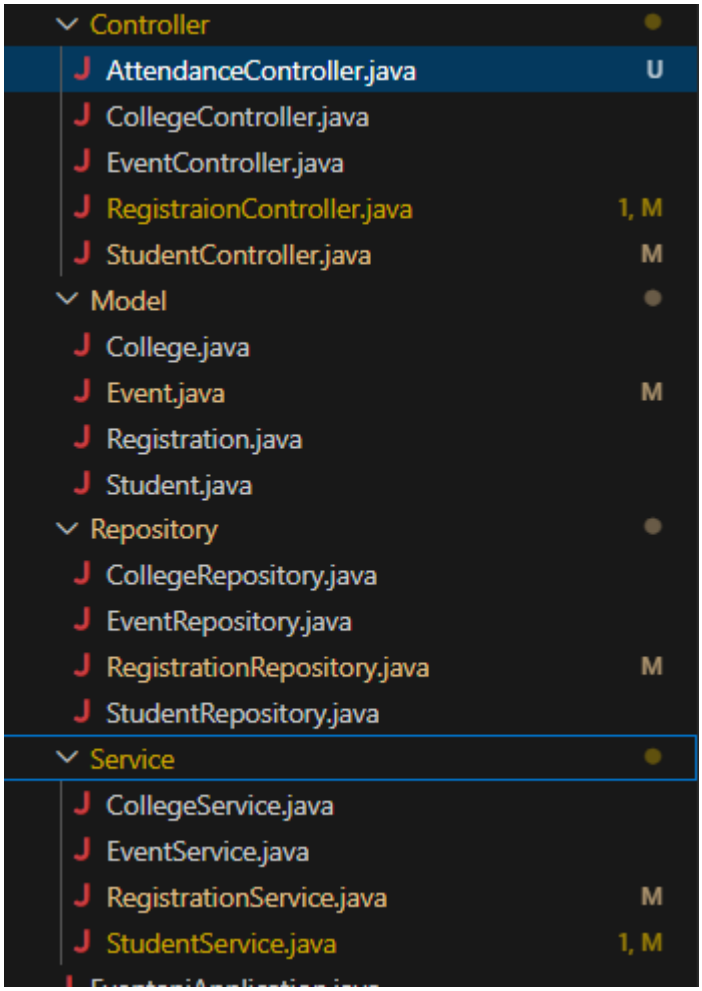


Giridharan KS ENG22CS0306



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

Giridharan KS ENG22CS0306



File Structure

Models(Entity):

College:

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

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

Giridharan KS ENG22CS0306

Event:

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

Registration:

```java
@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";


}
```

Giridharan KS ENG22CS0306

Student:

```java
@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:

```java
@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;

    }
}
```

Giridharan KS ENG22CS0306

Event:

```java
@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);
        }
    }
}
```

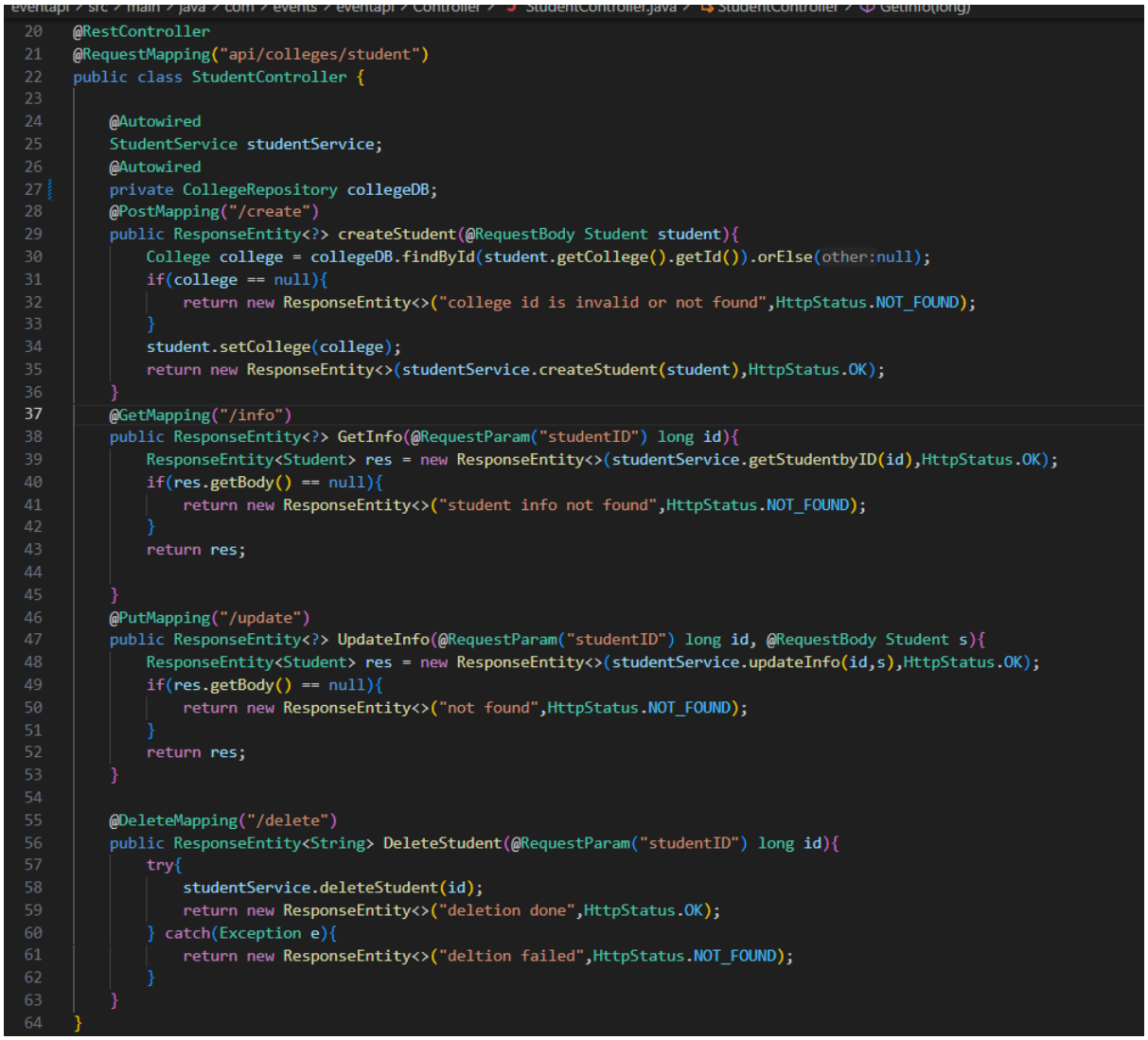
Giridharan KS ENG22CS0306

Registration:

```java
@RestController
@RequestMapping("api/colleges/event/regis")

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);
    }
}
```

Giridharan KS ENG22CS0306

Student:



```java
@RestController
@RequestMapping("api/colleges/student")
public class StudentController {

    @Autowired
    StudentService studentService;
    @Autowired
    private CollegeRepository collegeDB;
    @PostMapping("/create")
    public ResponseEntity<?> createStudent(@RequestBody Student student){
        College college = collegeDB.findById(student.getCollege().getId()).orElse(other:null);
        if(college == null){
            return new ResponseEntity<>("college id is invalid or not found",HttpStatus.NOT_FOUND);
        }
        student.setCollege(college);
        return new ResponseEntity<>(studentService.createStudent(student),HttpStatus.OK);
    }
    @GetMapping("/info")
    public ResponseEntity<?> GetInfo(@RequestParam("studentID") long id){
        ResponseEntity<Student> res = new ResponseEntity<>(studentService.getStudentbyID(id),HttpStatus.OK);
        if(res.getBody() == null){
            return new ResponseEntity<>("student info not found",HttpStatus.NOT_FOUND);
        }
        return res;

    }
    @PutMapping("/update")
    public ResponseEntity<?> UpdateInfo(@RequestParam("studentID") long id, @RequestBody Student s){
        ResponseEntity<Student> res = new ResponseEntity<>(studentService.updateInfo(id,s),HttpStatus.OK);
        if(res.getBody() == null){
            return new ResponseEntity<>("not found",HttpStatus.NOT_FOUND);
        }
        return res;
    }

    @DeleteMapping("/delete")
    public ResponseEntity<String> DeleteStudent(@RequestParam("studentID") long id){
        try{
            studentService.deleteStudent(id);
            return new ResponseEntity<>("deletion done",HttpStatus.OK);
        } catch(Exception e){
            return new ResponseEntity<>("deltion failed",HttpStatus.NOT_FOUND);
        }
    }
}
```

Giridharan KS ENG22CS0306

Attendance:

```java
@RestController
@RequestMapping("/api/attendance")
public class AttendanceController {
    @Autowired
    private RegistrationRepository registrationRepo;
    @PutMapping("/mark/{regId}/")
    public ResponseEntity<?> markAttendance(@PathVariable Long regId,
                                            @RequestParam String status) {
        Registration reg = registrationRepo.findById(regId).orElse(other:null);
        if (reg == null) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Registration not found");
        }

        if (!status.equalsIgnoreCase(anotherString:"present") && !status.equalsIgnoreCase(anotherString:"absent")) {
            return ResponseEntity.badRequest().body("Status must be 'present' or 'absent'");
        }

        reg.setAttendance(status.toLowerCase());
        registrationRepo.save(reg);

        return ResponseEntity.ok("Attendance marked as " + status);
    }

    @GetMapping("/summary/{eventId}")
    public ResponseEntity<?> getEventSummary(@PathVariable Long eventId) {
        long total = registrationRepo.countByEventEventid(eventId);
        long present = registrationRepo.countByEventEventidAndAttendance(eventId, attendance:"present");
        long absent = registrationRepo.countByEventEventidAndAttendance(eventId, attendance:"absent");

        Map<String, Object> summary = new HashMap<>();
        summary.put(key:"total_registered", total);
        summary.put(key:"present_count", present);
        summary.put(key:"absent_count", absent);

        return ResponseEntity.ok(summary);
    }

    @GetMapping("/present/{eventId}")
    public ResponseEntity<?> getPresentList(@PathVariable Long eventId) {
        return ResponseEntity.ok(registrationRepo.findByEventEventidAndAttendance(eventId, attendance:"present"));
    }

    @GetMapping("/absent/{eventId}")
    public ResponseEntity<?> getAbsentList(@PathVariable Long eventId) {
        return ResponseEntity.ok(registrationRepo.findByEventEventidAndAttendance(eventId, attendance:"absent"));
```

Giridharan KS ENG22CS0306

Repositories:

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>{

}
```

Giridharan KS ENG22CS0306

Service:

College:

```java
@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);

    }

}
```

Giridharan KS ENG22CS0306

Event:

```java
@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> getallEventsbbyid(long id){
        return EventDB.findByCollegeId(id);
    }

    public Event getEventByid(long id){
        return EventDB.findById(id).orElse(other: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);
    }

}
```

Giridharan KS ENG22CS0306

Registration:

```java
@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);
    }
}
```

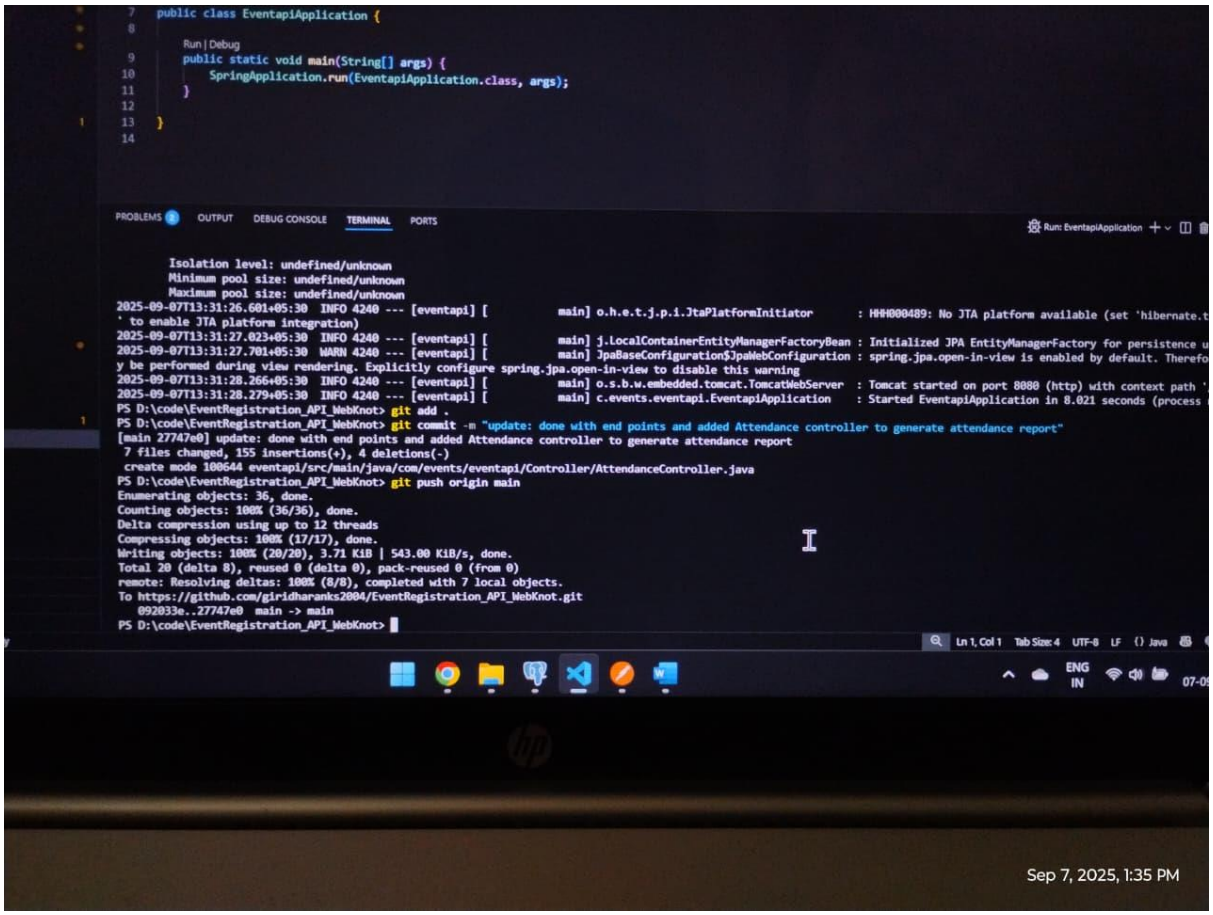
Giridharan KS ENG22CS0306

–Student:

```java
@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);
    }

}
```

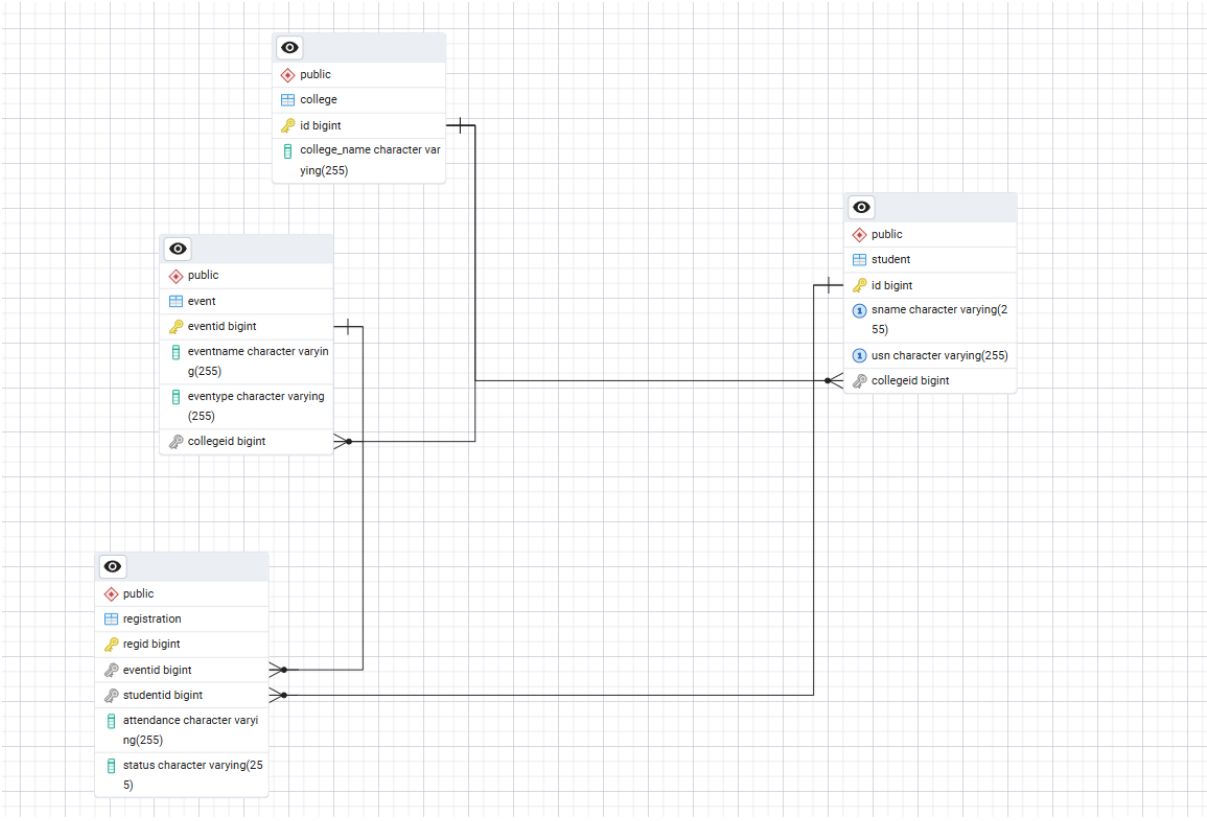
For controllers do check:
https://github.com/giridharanks2004/EventRegistration_API_WebKnot

Giridharan KS ENG22CS0306



Final push before adding documentations

Giridharan KS ENG22CS0306



ER Diagram Postgresql