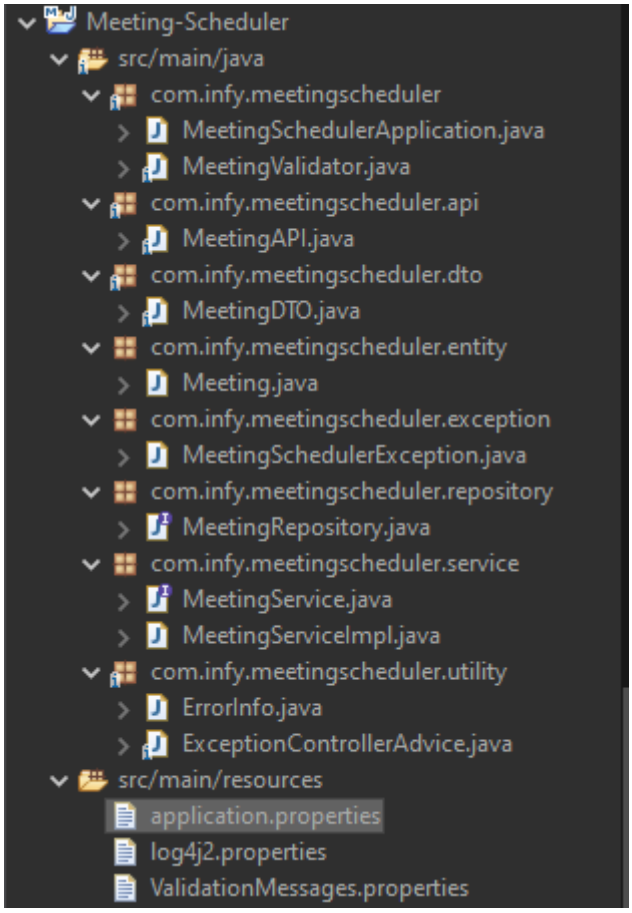
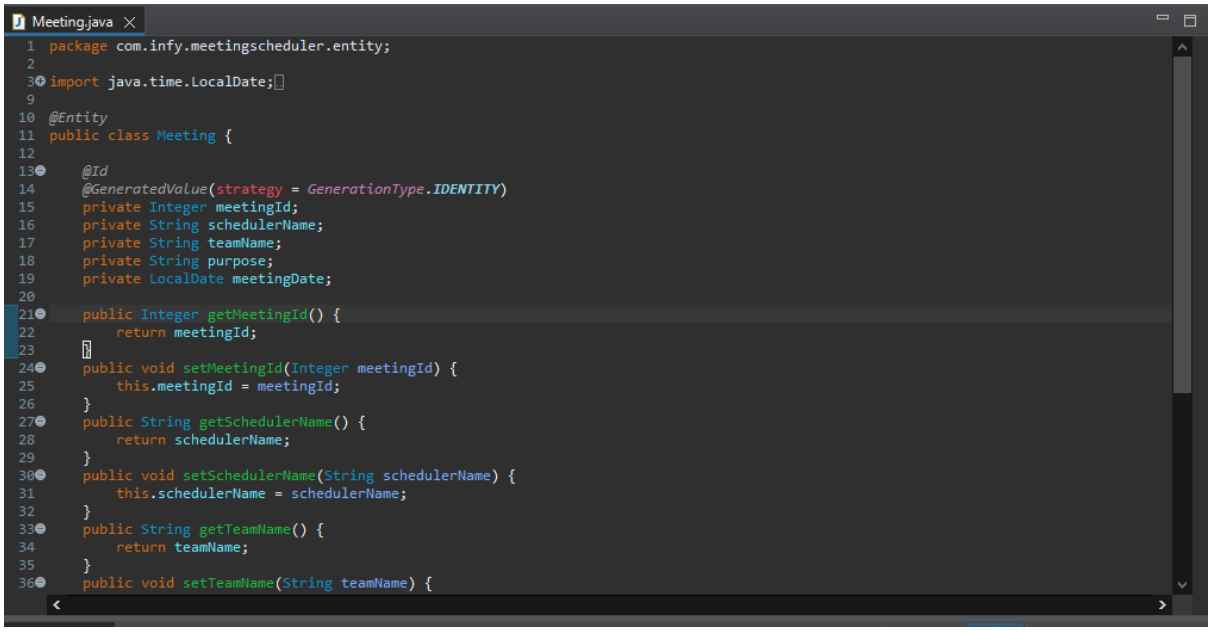
MEETING SCHEDULER

Project Structure:



Meeting.java (Entity Class), for this, compare with table script (SQL File) and if table name varies use @Table & for Columns use @Column (If column name varies) -> To be modified



Meeting.java

```
    }
    public void setMeetingId(Integer meetingId) {
        this.meetingId = meetingId;
    }
    public String getSchedulerName() {
        return schedulerName;
    }
    public void setSchedulerName(String schedulerName) {
        this.schedulerName = schedulerName;
    }
    public String getTeamName() {
        return teamName;
    }
    public void setTeamName(String teamName) {
        this.teamName = teamName;
    }
    public String getPurpose() {
        return purpose;
    }
    public void setPurpose(String purpose) {
        this.purpose = purpose;
    }
    public LocalDate getMeetingDate() {
        return meetingDate;
    }
    public void setMeetingDate(LocalDate meetingDate) {
        this.meetingDate = meetingDate;
    }

}

```

MeetingDTO.java (DTO Class), for validation, messages are implemented in ValidationMessages.properties -> to be modified

MeetingDTO.java

```
package com.infy.meetingscheduler.dto;

import java.time.LocalDate;

//Here for Validation, messages are refereed from ValidationMessages.properties
public class MeetingDTO {

    private Integer meetingId;

    @NotNull(message = "{meeting.scheduler.absent}")
    @Pattern(regexp = "([a-zA-Z]+)(\\s[a-zA-Z]+)*", message = "{meeting.scheduler.invalid}")
    private String schedulerName;

    @NotNull(message = "{meeting.team.absent}")
    private String teamName;

    @NotNull(message = "{meeting.purpose.absent}")
    private String purpose;

    @NotNull(message = "{meeting.date.absent}")
    @FutureOrPresent(message = "{meeting.date.invalid}")
    private LocalDate meetingDate;

    public Integer getMeetingId() {
        return meetingId;
    }
    public void setMeetingId(Integer meetingId) {
        this.meetingId = meetingId;
    }
    public String getSchedulerName() {
        return schedulerName;
```

MeetingDTO.java

```java
    public String getTeamName() {
        return teamName;
    }
    public void setTeamName(String teamName) {
        this.teamName = teamName;
    }
    public String getPurpose() {
        return purpose;
    }
    public void setPurpose(String purpose) {
        this.purpose = purpose;
    }
    public LocalDate getMeetingDate() {
        return meetingDate;
    }
    public void setMeetingDate(LocalDate meetingDate) {
        this.meetingDate = meetingDate;
    }

    // This code may vary in assessment and there is no need to code this as this is already provided
    public MeetingDTO prepareDTO(Meeting meeting) {
        MeetingDTO meetingDTO = new MeetingDTO();
        meetingDTO.setMeetingDate(meeting.getMeetingDate());
        meetingDTO.setMeetingId(meeting.getMeetingId());
        meetingDTO.setPurpose(meeting.getPurpose());
        meetingDTO.setSchedulerName(meeting.getSchedulerName());
        meetingDTO.setTeamName(meeting.getTeamName());
        return meetingDTO;
    }

    // This code may vary in assessment and there is no need to code this as this is already provided
```

```java
    // This code may vary in assessment and there is no need to code this as this is already provided
    public Meeting prepareEntity() {
        Meeting meeting = new Meeting();
        meeting.setMeetingDate(this.getMeetingDate());
        meeting.setPurpose(this.getPurpose());
        meeting.setSchedulerName(this.getSchedulerName());
        meeting.setTeamName(this.getTeamName());
        return meeting;
    }

}
```

MeetingValidator.java (Validator class), for constructor, code is already present -> To be Implemented

MeetingValidator.java

```java
package com.infy.meetingscheduler;

import java.time.DayOfWeek;

public class MeetingValidator {

    // This Code is already implemented as per the question, so dont modify constructor code
    private MeetingValidator() {
    // body already implemented as per question
  }

    public static void validateMeeting(MeetingDTO meetingDTO) throws MeetingSchedulerException{
        if(Boolean.FALSE.equals(isValidTeamName(meetingDTO.getTeamName()))) {
            throw new MeetingSchedulerException("MeetingValidator.INVALID_TEAM_NAME");
        }
        if(Boolean.FALSE.equals(isValidMeetingDate(meetingDTO.getMeetingDate()))) {
            throw new MeetingSchedulerException("MeetingValidator.INVALID_MEETING_DATE");
        }
    }

    public static Boolean isValidTeamName(String teamName) {
         String regex = "ETAMYS(JAVA|UI|BI|MS|AI)";
         return teamName.matches(regex);
    }

    public static Boolean isValidMeetingDate(LocalDate meetingDate) {
return Boolean.FALSE.equals((meetingDate.getDayOfWeek().equals(DayOfWeek.SATURDAY) || meetingDate.getDayOfWeek().equals(DayOfWeek.SUNDA
    }

}
```

MeetingRepository.java (Repository Interface) -> To be Implemented

MeetingRepository.java

```java
package com.infy.meetingscheduler.repository;

import java.time.LocalDate;

public interface MeetingRepository extends CrudRepository<Meeting, Integer>{

//  @Query("SELECT m FROM Meeting m WHERE m.schedulerName = ?1")
    List<Meeting> findBySchedulerName(String schedulerName);

//  @Query("SELECT m FROM Meeting m WHERE m.schedulerName = ?1 AND m.meetingDate = ?2")
    List<Meeting> findBySchedulerNameAndMeetingDate(String schedulerName, LocalDate meetingDate);

//  @Query("SELECT m FROM Meeting m WHERE m.teamName = ?1 AND m.meetingDate = ?2")
    List<Meeting> findByTeamNameAndMeetingDate(String teamName, LocalDate meetingDate);

}

```

MeetingService.java (Service Interface) -> Already Implemented

MeetingService.java

```java
package com.infy.meetingscheduler.service;

import java.util.List;

public interface MeetingService {

    List<MeetingDTO> getAllMeetingsOfScheduler(String schedulerName) throws MeetingSchedulerException;
    MeetingDTO scheduleMeeting(MeetingDTO meetingDTO) throws MeetingSchedulerException;

}
```

MeetingServiceImpl.java (Service Implementation Class) -> To be Implemented

MeetingServiceImpl.java

```java
package com.infy.meetingscheduler.service;

import java.util.ArrayList;

@Service("meetingService")
@Transactional
public class MeetingServiceImpl implements MeetingService{

    @Autowired
    private MeetingRepository meetingRepository;

    @Override
    public List<MeetingDTO> getAllMeetingsOfScheduler(String schedulerName) throws MeetingSchedulerException{
        List<Meeting> meetingList=meetingRepository.findBySchedulerName(schedulerName);
        if(meetingList.isEmpty())
            throw new MeetingSchedulerException("MeetingService.NO_MEETINGS_FOUND");
        List<MeetingDTO> meetingDTOList = new ArrayList<>();
        meetingList.
        stream().
        forEach(x-> meetingDTOList.add(new MeetingDTO().prepareDTO(x)));
        meetingDTOList.sort((x1,x2)->x1.getMeetingDate().compareTo(x2.getMeetingDate()));
        return meetingDTOList;
    }

    @Override
    public MeetingDTO scheduleMeeting(MeetingDTO meetingDTO) throws MeetingSchedulerException{
        MeetingValidator.validateMeeting(meetingDTO);
        List<Meeting> meetingList = meetingRepository.findBySchedulerNameAndMeetingDate(meetingDTO.getSchedulerName(),
                meetingDTO.getMeetingDate());
        if(Boolean.FALSE.equals(meetingList.isEmpty()))
            throw new MeetingSchedulerException("MeetingService.MEETING_DATE_UNAVAILABLE");
```

MeetingServiceImpl.java

```java
    @Override
    public List<MeetingDTO> getAllMeetingsOfScheduler(String schedulerName) throws MeetingSchedulerException{
        List<Meeting> meetingList=meetingRepository.findBySchedulerName(schedulerName);
        if(meetingList.isEmpty())
            throw new MeetingSchedulerException("MeetingService.NO_MEETINGS_FOUND");
        List<MeetingDTO> meetingDTOList = new ArrayList<>();
        meetingList.
        stream().
        forEach(x-> meetingDTOList.add(new MeetingDTO().prepareDTO(x)));
        meetingDTOList.sort((x1,x2)->x1.getMeetingDate().compareTo(x2.getMeetingDate()));
        return meetingDTOList;
    }

    @Override
    public MeetingDTO scheduleMeeting(MeetingDTO meetingDTO) throws MeetingSchedulerException{
        MeetingValidator.validateMeeting(meetingDTO);
        List<Meeting> meetingList = meetingRepository.findBySchedulerNameAndMeetingDate(meetingDTO.getSchedulerName(),
                meetingDTO.getMeetingDate());
        if(Boolean.FALSE.equals(meetingList.isEmpty()))
            throw new MeetingSchedulerException("MeetingService.MEETING_DATE_UNAVAILABLE");
        List<Meeting> teamMeetingList=meetingRepository.findByTeamNameAndMeetingDate(meetingDTO.getTeamName(), meetingDTO.getMeetingDat
        if(Boolean.FALSE.equals(teamMeetingList.isEmpty()))
            throw new MeetingSchedulerException("MeetingService.TEAM_UNAVAILABLE");
        Integer meetingId=meetingRepository.save(meetingDTO.prepareEntity()).getMeetingId();
        meetingDTO.setMeetingId(meetingId);
        return meetingDTO;
    }

}
```

MeetingAPI (API Implementation Class), for validation, messages is picked up from ValidationMessages.properties -> to be implemented

MeetingAPI.java

```java
package com.infy.meetingscheduler.api;

import java.util.List;

@RestController
@RequestMapping("/api")
@Validated
public class MeetingAPI {

    @Autowired
    private MeetingService meetingService;

    // Here for Validation, messages are refereed from ValidationMessages.properties
    @GetMapping("/meetings/{schedulerName}")
    public ResponseEntity<List<MeetingDTO>> getAllMeetingOfScheduler(
            @PathVariable
            @Pattern(regexp = "([a-zA-Z]+)(\\s[a-zA-Z]+)*", message = "{meeting.scheduler.invalid}")
            String schedulerName
            ) throws MeetingSchedulerException {
        List<MeetingDTO> meetingList=meetingService.getAllMeetingsOfScheduler(schedulerName);
        return new ResponseEntity<>(meetingList, HttpStatus.OK);
    }

    @PostMapping("/meetings")
    public ResponseEntity<MeetingDTO> scheduleMeeting(@Valid @RequestBody MeetingDTO meetingDetails) throws MeetingSchedulerException {
        MeetingDTO meetingObject=meetingService.scheduleMeeting(meetingDetails);
        return new ResponseEntity<>(meetingObject, HttpStatus.CREATED);
    }

}
```

ErrorInfo.java -> Already Implemented

ErrorInfo.java

```java
package com.infy.meetingscheduler.utility;

public class ErrorInfo {

    private String errorMessage;
    private Integer errorCode;

    public String getErrorMessage() {
        return errorMessage;
    }

    public void setErrorMessage(String errorMessage) {
        this.errorMessage = errorMessage;
    }

    public Integer getErrorCode() {
        return errorCode;
    }

    public void setErrorCode(Integer errorCode) {
        this.errorCode = errorCode;
    }

}

```

ExceptionControllerAdvice.java (Exception handler class) for these usually methods will be implemented, the focus should be on annotations -> to be modified

ExceptionControllerAdvice.java

```java
package com.infy.meetingscheduler.utility;

import java.util.stream.Collectors;

// For this, usually methods will be already implemented, so focus should be on Annotations
@RestControllerAdvice
public class ExceptionControllerAdvice {

    @Autowired
    private Environment env;

    private static final Log LOGGER = LogFactory.getLog(ExceptionControllerAdvice.class);

    @ExceptionHandler(MeetingSchedulerException.class)
    public ResponseEntity<ErrorInfo> meetingSchedulerExceptionHandler(MeetingSchedulerException exception){
        LOGGER.error(exception.getMessage(), exception);
        ErrorInfo errorInfo = new ErrorInfo();
        errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
        errorInfo.setErrorMessage(env.getProperty(exception.getMessage()));
        return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
    }

    @ExceptionHandler(Exception.class)
    public ResponseEntity<ErrorInfo> generalExceptionHandler(Exception exception)
    {
    LOGGER.error(exception.getMessage(), exception);
    ErrorInfo errorInfo = new ErrorInfo();
    errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
    errorInfo.setErrorMessage(env.getProperty("General.EXCEPTION_MESSAGE"));
    return new ResponseEntity<>(errorInfo,
                    HttpStatus.INTERNAL_SERVER_ERROR);
```

ExceptionControllerAdvice.java

```
    errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
    errorInfo.setErrorMessage(env.getProperty("General.EXCEPTION_MESSAGE"));
    return new ResponseEntity<>(errorInfo,
                    HttpStatus.INTERNAL_SERVER_ERROR);
    }

    @ExceptionHandler({MethodArgumentNotValidException.class, ConstraintViolationException.class})
    public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception)
    {
    LOGGER.error(exception.getMessage(), exception);
    String errorMsg;
    if (exception instanceof MethodArgumentNotValidException)
    {
        MethodArgumentNotValidException manvException = (MethodArgumentNotValidException) exception;
        errorMsg = manvException.getBindingResult()
                    .getAllErrors()
                    .stream()
                    .map(ObjectError::getDefaultMessage)
                    .collect(Collectors.joining(", "));

    }
    else
    {
        ConstraintViolationException cvException = (ConstraintViolationException) exception;
        errorMsg = cvException.getConstraintViolations()
                  .stream()
                  .map(ConstraintViolation::getMessage)
                  .collect(Collectors.joining(", "));

    }
    ErrorInfo errorInfo = new ErrorInfo();
```

ExceptionControllerAdvice.java

```
    public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception)
    {
    LOGGER.error(exception.getMessage(), exception);
    String errorMsg;
    if (exception instanceof MethodArgumentNotValidException)
    {
        MethodArgumentNotValidException manvException = (MethodArgumentNotValidException) exception;
        errorMsg = manvException.getBindingResult()
                    .getAllErrors()
                    .stream()
                    .map(ObjectError::getDefaultMessage)
                    .collect(Collectors.joining(", "));

    }
    else
    {
        ConstraintViolationException cvException = (ConstraintViolationException) exception;
        errorMsg = cvException.getConstraintViolations()
                  .stream()
                  .map(ConstraintViolation::getMessage)
                  .collect(Collectors.joining(", "));

    }
    ErrorInfo errorInfo = new ErrorInfo();
    errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
    errorInfo.setErrorMessage(errorMsg);
    return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
    }
}
```

MeetingSchedulerApplication.java (Main class) -> Already implemented

MeetingSchedulerApplication.java

```
package com.infy.meetingscheduler;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class MeetingSchedulerApplication {

    public static void main(String[] args) {
        SpringApplication.run(MeetingSchedulerApplication.class, args);
    }

}
```

MeetingSchedulerException.java (Exception Class) -> Already Implemented

MeetingSchedulerException.java

```java
package com.infy.meetingscheduler.exception;

// Already Implemented
public class MeetingSchedulerException extends Exception{


    private static final long serialVersionUID = 1L;

    public MeetingSchedulerException(String message) {
        super(message);
    }

}
```

Application.properties (Properties file) -> Already implemented

application.properties

```
MeetingValidator.INVALID_TEAM_NAME=Team Name should be one among ETAMYSJAVA, ETAMYSMS, ETAMYSBI, ETAMYSUI, ETAMYSAI
MeetingValidator.INVALID_MEETING_DATE=Meeting Date cannot be a Saturday or a Sunday
MeetingService.NO_MEETINGS_FOUND=No Meetings were found to the given Scheduler
MeetingService.MEETING_DATE_UNAVAILABLE=The scheduler has a meeting on the requested date, please choose any other date
MeetingService.TEAM_UNAVAILABLE=The team has a meeting on the requested date, please choose any other date
General.EXCEPTION_MESSAGE=Internal Server Error!

# Database properties
spring.datasource.url=jdbc:mysql://localhost:3306/infymeeting_db
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

server.port=8765
```

ValidationMessages.properties (Validation messages file) -> Already implemented

ValidationMessages.properties

```
meeting.scheduler.invalid=Scheduler name should have only letters and spaces
meeting.scheduler.absent=Scheduler name is mandatory
meeting.team.absent=Team name is mandatory
meeting.purpose.absent=Purpose is mandatory
meeting.date.absent=Meeting date is mandatory
meeting.date.invalid=Meeting date should not be a past date
```

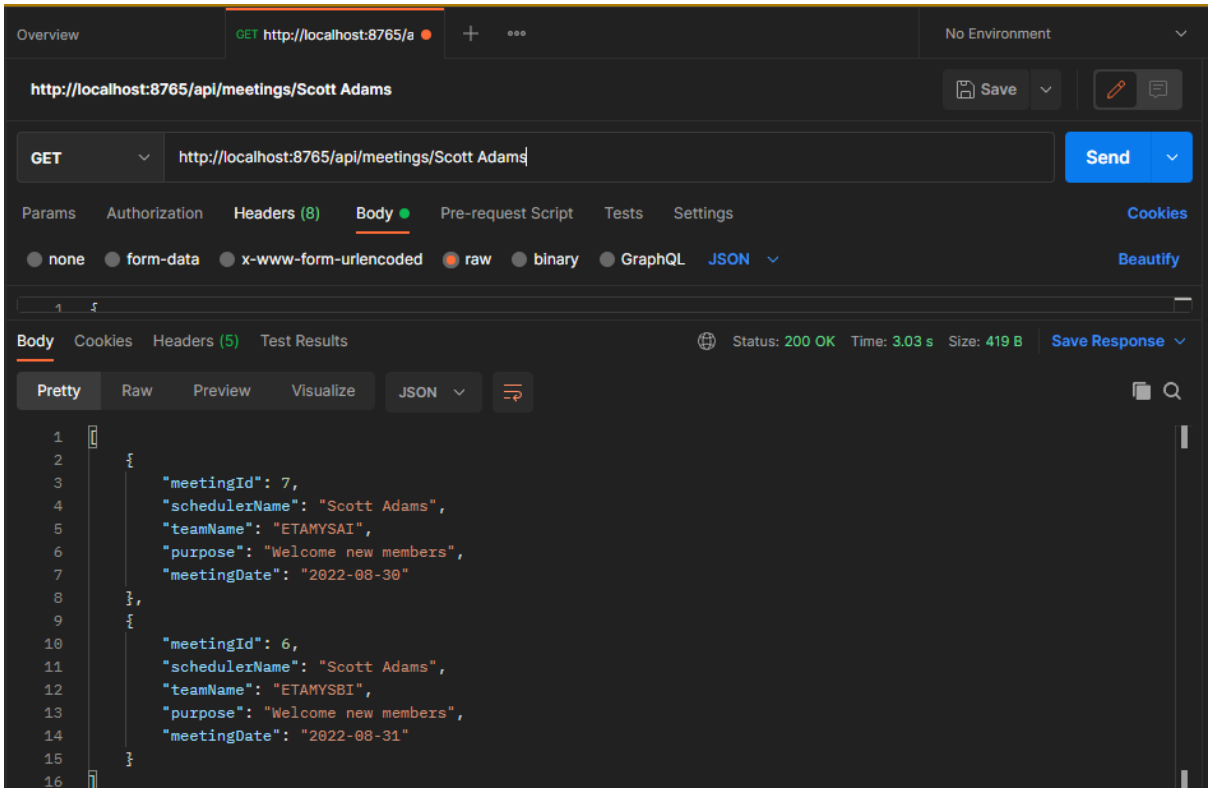
Log4j2.properties (Logger config file) -> Already Implemented

log4j2.properties

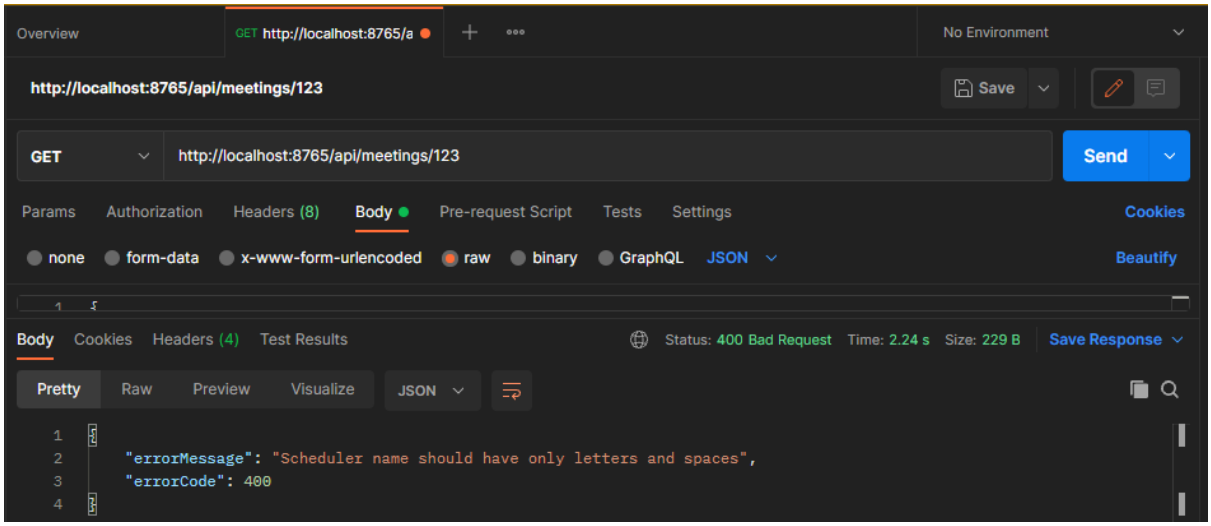
```
name=LoggingFile
rootLogger.level=info
rootLogger.appenderRef.file.ref=LoggerAppender

appender.file.type=File
appender.file.name=LoggerAppender
appender.file.fileName=log/ErrorLog.log
appender.file.layout.type=PatternLayout
appender.file.layout.pattern=%d{dd-MMM-yyyy HH:mm:ss} %level - %m%n

# Console Appender
appender.console.name=ConsoleAppender
appender.console.type=Console
appender.console.layout.type=PatternLayout
appender.console.layout.pattern=%m%n


#Declaring logger for business logic
logger.infyacademy.name=com.infy.meetingscheduler.utility
logger.infyacademy.level=DEBUG
logger.infyacademy.appenderRef.file.ref=LoggerAppender
logger.infyacademy.additivity=false

logger.tester.name=com.infy.meetingscheduler
logger.tester.level=INFO
logger.tester.appenderRef.file.ref=ConsoleAppender
logger.tester.additivity=false
```

API Tests (Refer QP for Test Cases) -> To be done in postman

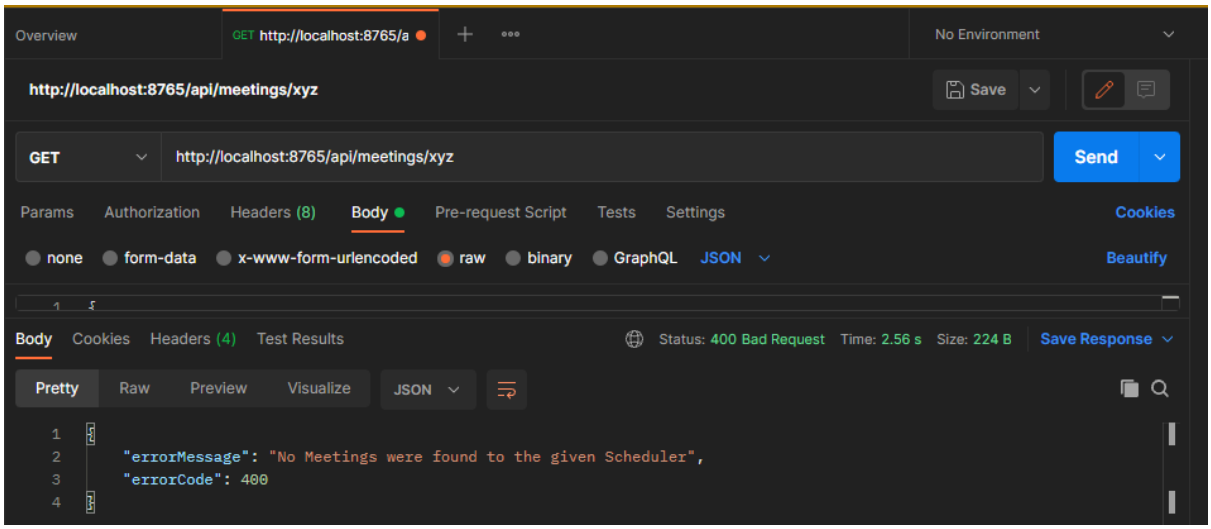
1. GET: Valid Test Data



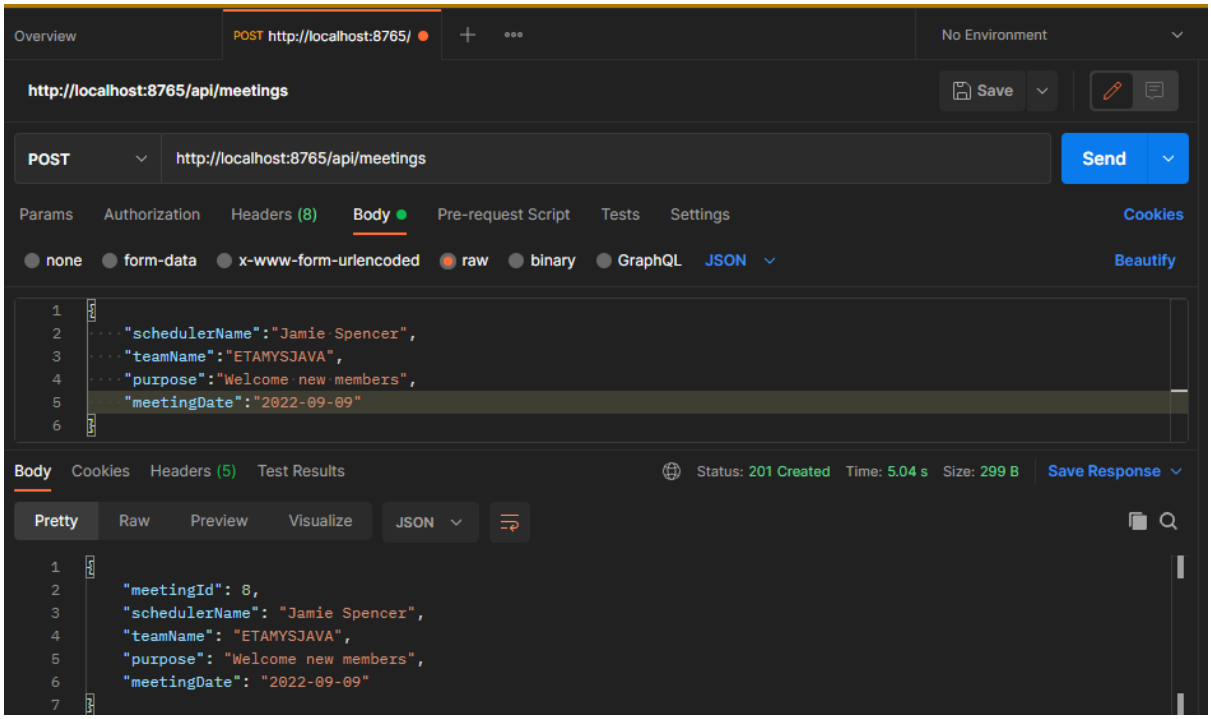
2. GET: Invalid Test Data 1



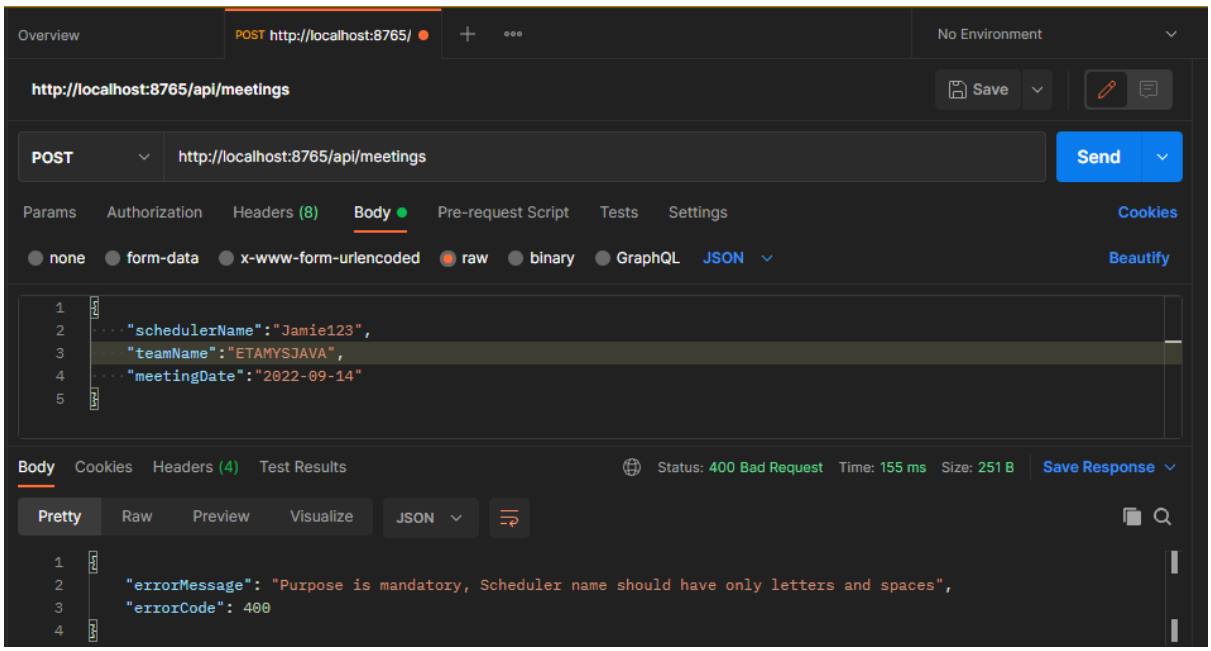
3. GET: Invalid Test Data 2



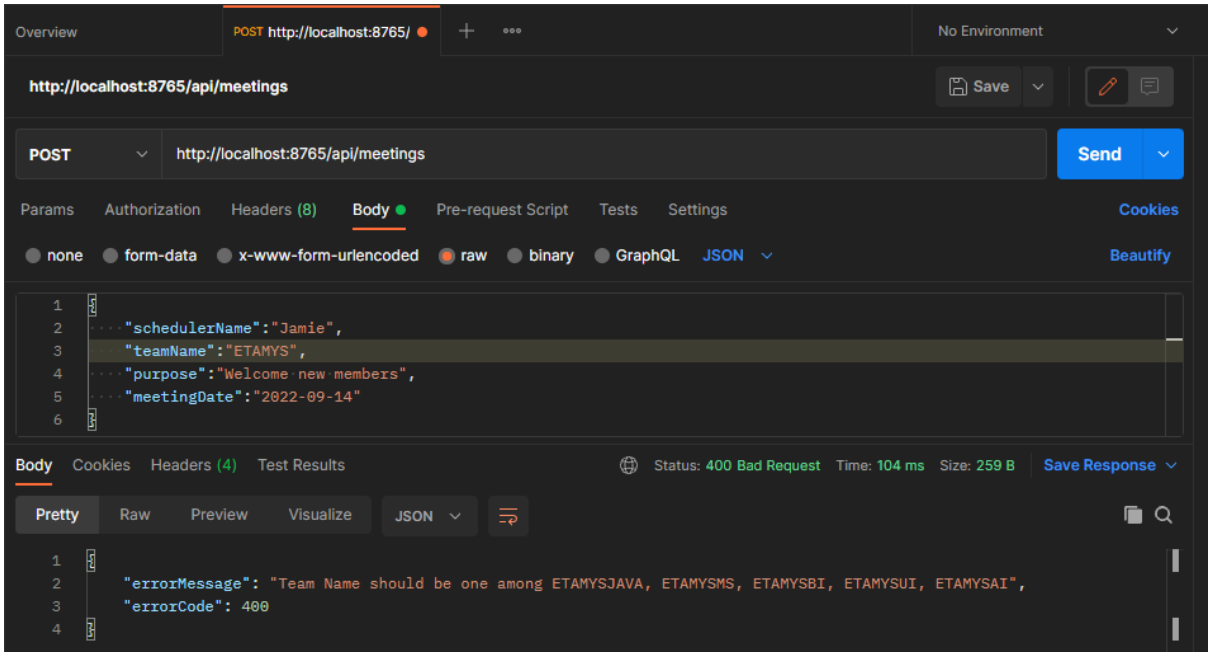
4. POST: Valid Test Data



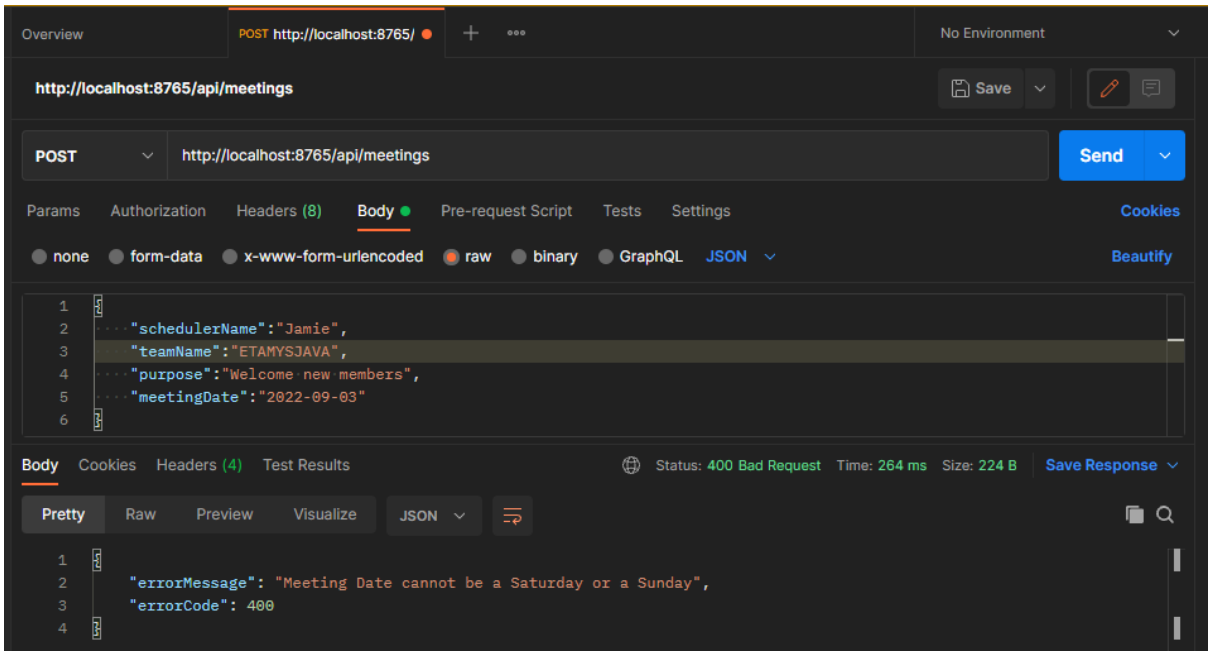
5. POST: Invalid Test Data 1



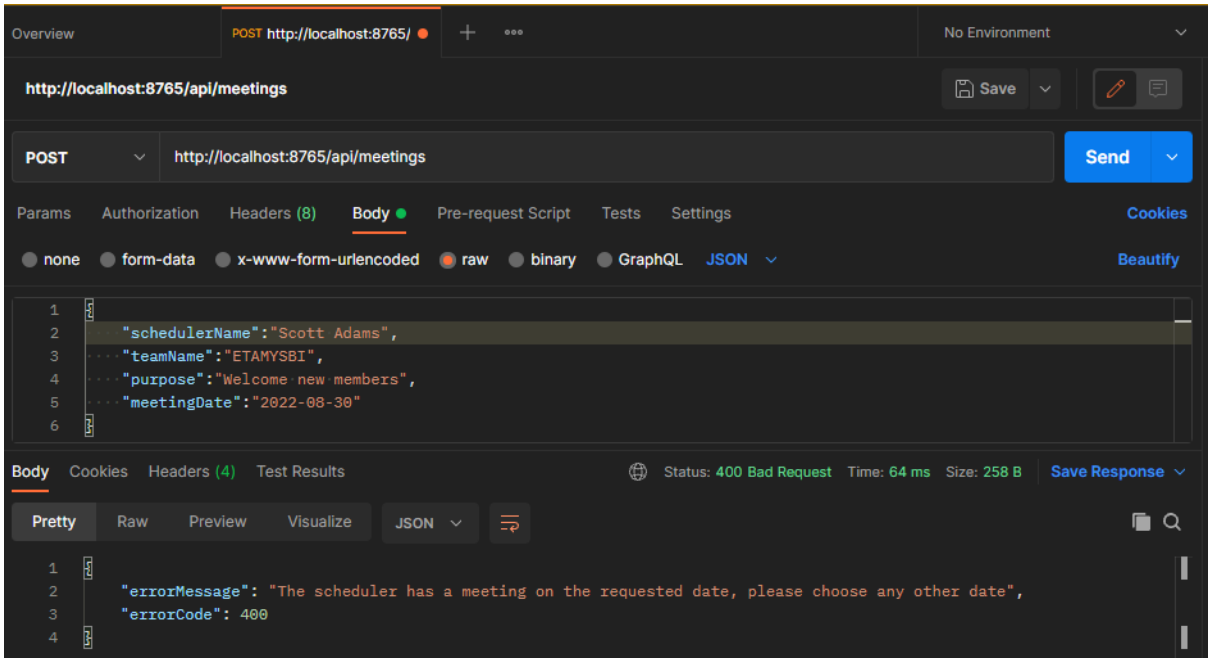
6. POST: Invalid Test Data 2



7. POST: Invalid Test Data 3



8. POST: Invalid Test Data 4



9. POST: Invalid Test Data 5