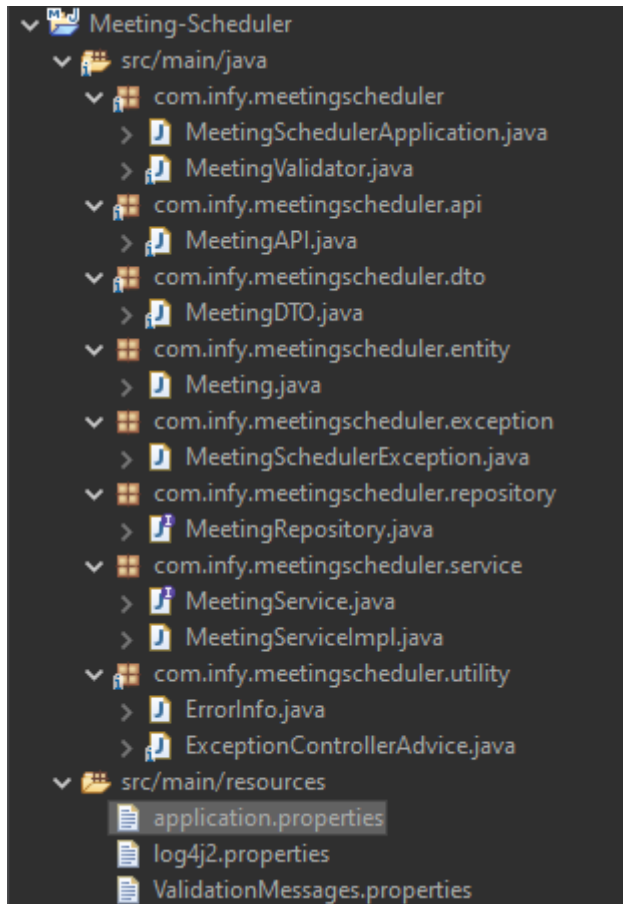
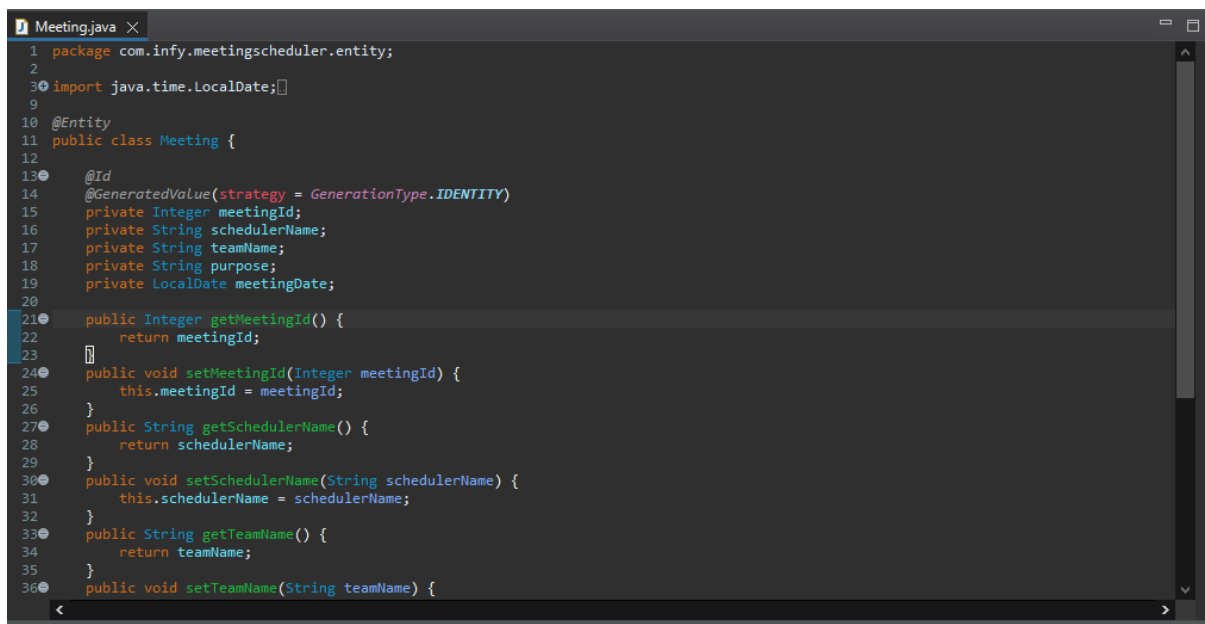


MEETING SCHEDULER

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 X
23
24 public void setMeetingId(Integer meetingId) {
25     this.meetingId = meetingId;
26 }
27 public String getSchedulderName() {
28     return schedulderName;
29 }
30 public void setSchedulderName(String schedulderName) {
31     this.schedulderName = schedulderName;
32 }
33 public String getTeamName() {
34     return teamName;
35 }
36 public void setTeamName(String teamName) {
37     this.teamName = teamName;
38 }
39 public String getPurpose() {
40     return purpose;
41 }
42 public void setPurpose(String purpose) {
43     this.purpose = purpose;
44 }
45 public LocalDate getMeetingDate() {
46     return meetingDate;
47 }
48 public void setMeetingDate(LocalDate meetingDate) {
49     this.meetingDate = meetingDate;
50 }
51 }
52
53
```

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

```
MeetingDTO.java X
1 package com.infy.meetingscheduler.dto;
2
3 import java.time.LocalDate;
4
5 //Here for Validation, messages are refereed from ValidationMessages.properties
6 public class MeetingDTO {
7
8     private Integer meetingId;
9
10     @NotNull(message = "{meeting.schedulder.absent}")
11     @Pattern(regexp = "([a-zA-Z]+)(\\s[a-zA-Z]+)*", message = "{meeting.schedulder.invalid}")
12     private String schedulderName;
13
14     @NotNull(message = "{meeting.team.absent}")
15     private String teamName;
16
17     @NotNull(message = "{meeting.purpose.absent}")
18     private String purpose;
19
20     @NotNull(message = "{meeting.date.absent}")
21     @FutureOrPresent(message = "{meeting.date.invalid}")
22     private LocalDate meetingDate;
23
24     public Integer getMeetingId() {
25         return meetingId;
26     }
27     public void setMeetingId(Integer meetingId) {
28         this.meetingId = meetingId;
29     }
30     public String getSchedulderName() {
31         return schedulderName;
32     }
33 }
```

```
MeetingDTO.java X
42 public String getTeamName() {
43     return teamName;
44 }
45 public void setTeamName(String teamName) {
46     this.teamName = teamName;
47 }
48 public String getPurpose() {
49     return purpose;
50 }
51 public void setPurpose(String purpose) {
52     this.purpose = purpose;
53 }
54 public LocalDate getMeetingDate() {
55     return meetingDate;
56 }
57 public void setMeetingDate(LocalDate meetingDate) {
58     this.meetingDate = meetingDate;
59 }
60
61 // This code may vary in assessment and there is no need to code this as this is already provided
62 public MeetingDTO prepareDTO(Meeting meeting) {
63     MeetingDTO meetingDTO = new MeetingDTO();
64     meetingDTO.setMeetingDate(meeting.getMeetingDate());
65     meetingDTO.setMeetingId(meeting.getMeetingId());
66     meetingDTO.setPurpose(meeting.getPurpose());
67     meetingDTO.setSchedulerName(meeting.getSchedulerName());
68     meetingDTO.setTeamName(meeting.getTeamName());
69     return meetingDTO;
70 }
71
72 // This code may vary in assessment and there is no need to code this as this is already provided
```

```
// 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 X
1 package com.infy.meetingscheduler;
2
3 import java.time.DayOfWeek;
4
5
6
7
8
9 public class MeetingValidator {
10
11     // This Code is already implemented as per the question, so don't modify constructor code
12     private MeetingValidator() {
13         // body already implemented as per question
14     }
15
16     public static void validateMeeting(MeetingDTO meetingDTO) throws MeetingSchedulerException{
17         if(Boolean.FALSE.equals(isValidTeamName(meetingDTO.getTeamName()))){
18             throw new MeetingSchedulerException("MeetingValidator.INVALID_TEAM_NAME");
19         }
20         if(Boolean.FALSE.equals(isValidMeetingDate(meetingDTO.getMeetingDate()))){
21             throw new MeetingSchedulerException("MeetingValidator.INVALID_MEETING_DATE");
22         }
23     }
24
25     public static Boolean isValidTeamName(String teamName) {
26         String regex = "ETAMYS(JAVA|UI|BI|MS|AI)";
27         return teamName.matches(regex);
28     }
29
30     public static Boolean isValidMeetingDate(LocalDate meetingDate) {
31         return Boolean.FALSE.equals((meetingDate.getDayOfWeek().equals(DayOfWeek.SATURDAY) || meetingDate.getDayOfWeek().equals(DayOfWeek.SUNDAY)));
32     }
33
34 }
35
```

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

```
MeetingRepository.java X
1 package com.infy.meetingscheduler.repository;
2
3 import java.time.LocalDate;
4
5
6
7
8
9
10 public interface MeetingRepository extends CrudRepository<Meeting, Integer>{
11
12 // @Query("SELECT m FROM Meeting m WHERE m.schedulerName = ?1")
13 List<Meeting> findBySchedulerName(String schedulerName);
14
15 // @Query("SELECT m FROM Meeting m WHERE m.schedulerName = ?1 AND m.meetingDate = ?2")
16 List<Meeting> findBySchedulerNameAndMeetingDate(String schedulerName, LocalDate meetingDate);
17
18 // @Query("SELECT m FROM Meeting m WHERE m.teamName = ?1 AND m.meetingDate = ?2")
19 List<Meeting> findByTeamNameAndMeetingDate(String teamName, LocalDate meetingDate);
20
21 }
22
```

MeetingService.java (Service Interface) -> Already Implemented

```
MeetingService.java X
1 package com.infy.meetingscheduler.service;
2
3 import java.util.List;
4
5
6
7
8 public interface MeetingService {
9
10 List<MeetingDTO> getAllMeetingsOfScheduler(String schedulerName) throws MeetingSchedulerException;
11 MeetingDTO scheduleMeeting(MeetingDTO meetingDTO) throws MeetingSchedulerException;
12
13 }
14
```

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

```
MeetingServiceImpl.java X
1 package com.infy.meetingscheduler.service;
2
3 import java.util.ArrayList;
4
5
6
7
8
9
10
11
12
13
14
15
16
17 @Service("meetingService")
18 @Transactional
19 public class MeetingServiceImpl implements MeetingService{
20
21
22 @Autowired
23 private MeetingRepository meetingRepository;
24
25
26 @Override
27 public List<MeetingDTO> getAllMeetingsOfScheduler(String schedulerName) throws MeetingSchedulerException{
28 List<Meeting> meetingList=meetingRepository.findBySchedulerName(schedulerName);
29 if(meetingList.isEmpty())
30 throw new MeetingSchedulerException("MeetingService.NO_MEETINGS_FOUND");
31 List<MeetingDTO> meetingDTOList = new ArrayList<>();
32 meetingList.
33 stream().
34 forEach(x-> meetingDTOList.add(new MeetingDTO().prepareDTO(x)));
35 meetingDTOList.sort((x1,x2)->x1.getMeetingDate().compareTo(x2.getMeetingDate()));
36 return meetingDTOList;
37 }
38
39 @Override
40 public MeetingDTO scheduleMeeting(MeetingDTO meetingDTO) throws MeetingSchedulerException{
41 MeetingValidator.validateMeeting(meetingDTO);
42 List<Meeting> meetingList = meetingRepository.findBySchedulerNameAndMeetingDate(meetingDTO.getSchedulerName(),
43 meetingDTO.getMeetingDate());
44 if(Boolean.FALSE.equals(meetingList.isEmpty()))
45 throw new MeetingSchedulerException("MeetingService.MEETING_DATE_UNAVAILABLE");
46 }
47
```

```

23
24 @Override
25 public List<MeetingDTO> getAllMeetingsOfScheduler(String schedulerName) throws MeetingSchedulerException{
26     List<Meeting> meetingList=meetingRepository.findBySchedulerName(schedulerName);
27     if(meetingList.isEmpty())
28         throw new MeetingSchedulerException("MeetingService.NO_MEETINGS_FOUND");
29     List<MeetingDTO> meetingDTOList = new ArrayList<>();
30     meetingList.
31     stream().
32     forEach(x-> meetingDTOList.add(new MeetingDTO().prepareDTO(x)));
33     meetingDTOList.sort((x1,x2)->x1.getMeetingDate().compareTo(x2.getMeetingDate()));
34     return meetingDTOList;
35 }
36
37 @Override
38 public MeetingDTO scheduleMeeting(MeetingDTO meetingDTO) throws MeetingSchedulerException{
39     MeetingValidator.validateMeeting(meetingDTO);
40     List<Meeting> meetingList = meetingRepository.findBySchedulerNameAndMeetingDate(meetingDTO.getSchedulerName(),
41     meetingDTO.getMeetingDate());
42     if(Boolean.FALSE.equals(meetingList.isEmpty()))
43         throw new MeetingSchedulerException("MeetingService.MEETING_DATE_UNAVAILABLE");
44     List<Meeting> teamMeetingList=meetingRepository.findByTeamNameAndMeetingDate(meetingDTO.getTeamName(), meetingDTO.getMeetingDate());
45     if(Boolean.FALSE.equals(teamMeetingList.isEmpty()))
46         throw new MeetingSchedulerException("MeetingService.TEAM_UNAVAILABLE");
47     Integer meetingId=meetingRepository.save(meetingDTO.prepareEntity()).getMeetingId();
48     meetingDTO.setMeetingId(meetingId);
49     return meetingDTO;
50 }
51 }
52 }
53

```

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

```

1 package com.infy.meetingscheduler.api;
2
3 import java.util.List;
4
5 @RestController
6 @RequestMapping("/api")
7 @Validated
8 public class MeetingAPI {
9
10     @Autowired
11     private MeetingService meetingService;
12
13     // Here for Validation, messages are refereed from ValidationMessages.properties
14     @GetMapping("/meetings/{schedulerName}")
15     public ResponseEntity<List<MeetingDTO>> getAllMeetingOfScheduler(
16         @PathVariable
17         @Pattern(regexp = "[a-zA-Z]+(\\s[a-zA-Z]+)*", message = "{meeting.scheduler.invalid}")
18         String schedulerName
19     ) throws MeetingSchedulerException {
20         List<MeetingDTO> meetingList=meetingService.getAllMeetingsOfScheduler(schedulerName);
21         return new ResponseEntity<>(meetingList, HttpStatus.OK);
22     }
23
24     @PostMapping("/meetings")
25     public ResponseEntity<MeetingDTO> scheduleMeeting(@Valid @RequestBody MeetingDTO meetingDetails) throws MeetingSchedulerException {
26         MeetingDTO meetingObject=meetingService.scheduleMeeting(meetingDetails);
27         return new ResponseEntity<>(meetingObject, HttpStatus.CREATED);
28     }
29 }
30

```

ErrorInfo.java -> Already Implemented

```
1 package com.infy.meetingscheduler.utility;
2
3 public class ErrorInfo {
4
5     private String errorMessage;
6     private Integer errorCode;
7
8     public String getErrorMessage() {
9         return errorMessage;
10    }
11
12    public void setErrorMessage(String errorMessage) {
13        this.errorMessage = errorMessage;
14    }
15
16    public Integer getErrorCode() {
17        return errorCode;
18    }
19
20    public void setErrorCode(Integer errorCode) {
21        this.errorCode = errorCode;
22    }
23
24 }
25
```

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

```
1 package com.infy.meetingscheduler.utility;
2
3 import java.util.stream.Collectors;
4
5 // For this, usually methods will be already implemented, so focus should be on Annotations
6 @RestControllerAdvice
7 public class ExceptionControllerAdvice {
8
9     @Autowired
10     private Environment env;
11
12     private static final Log LOGGER = LoggerFactory.getLog(ExceptionControllerAdvice.class);
13
14     @ExceptionHandler(MeetingSchedulerException.class)
15     public ResponseEntity<ErrorInfo> meetingSchedulerExceptionHandler(MeetingSchedulerException exception){
16         LOGGER.error(exception.getMessage(), exception);
17         ErrorInfo errorInfo = new ErrorInfo();
18         errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
19         errorInfo.setErrorMessage(env.getProperty(exception.getMessage()));
20         return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
21     }
22
23     @ExceptionHandler(Exception.class)
24     public ResponseEntity<ErrorInfo> generalExceptionHandler(Exception exception)
25     {
26         LOGGER.error(exception.getMessage(), exception);
27         ErrorInfo errorInfo = new ErrorInfo();
28         errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
29         errorInfo.setErrorMessage(env.getProperty("General.EXCEPTION_MESSAGE"));
30         return new ResponseEntity<>(errorInfo,
31                                     HttpStatus.INTERNAL_SERVER_ERROR);
32     }
33 }
```

```
ExceptionHandlerAdvice.java X
44 errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
45 errorInfo.setErrorMessage(env.getProperty("General.EXCEPTION_MESSAGE"));
46 return new ResponseEntity<>(errorInfo,
47     HttpStatus.INTERNAL_SERVER_ERROR);
48 }
49
50 @ExceptionHandler({MethodArgumentNotValidException.class, ConstraintViolationException.class})
51 public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception)
52 {
53     LOGGER.error(exception.getMessage(), exception);
54     String errorMsg;
55     if (exception instanceof MethodArgumentNotValidException)
56     {
57         MethodArgumentNotValidException manvException = (MethodArgumentNotValidException) exception;
58         errorMsg = manvException.getBindingResult()
59             .getAllErrors()
60             .stream()
61             .map(ObjectError::getDefaultMessage)
62             .collect(Collectors.joining(", "));
63     }
64     }
65     else
66     {
67         ConstraintViolationException cvException = (ConstraintViolationException) exception;
68         errorMsg = cvException.getConstraintViolations()
69             .stream()
70             .map(ConstraintViolation::getMessage)
71             .collect(Collectors.joining(", "));
72     }
73     ErrorInfo errorInfo = new ErrorInfo();
74 }
```

```
ExceptionHandlerAdvice.java X
51 public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception)
52 {
53     LOGGER.error(exception.getMessage(), exception);
54     String errorMsg;
55     if (exception instanceof MethodArgumentNotValidException)
56     {
57         MethodArgumentNotValidException manvException = (MethodArgumentNotValidException) exception;
58         errorMsg = manvException.getBindingResult()
59             .getAllErrors()
60             .stream()
61             .map(ObjectError::getDefaultMessage)
62             .collect(Collectors.joining(", "));
63     }
64     }
65     else
66     {
67         ConstraintViolationException cvException = (ConstraintViolationException) exception;
68         errorMsg = cvException.getConstraintViolations()
69             .stream()
70             .map(ConstraintViolation::getMessage)
71             .collect(Collectors.joining(", "));
72     }
73     ErrorInfo errorInfo = new ErrorInfo();
74     errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
75     errorInfo.setErrorMessage(errorMsg);
76     return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
77 }
78 }
79 }
80 }
81 }
```

MeetingSchedulerApplication.java (Main class) -> Already implemented

```
MeetingSchedulerApplication.java X
1 package com.infy.meetingscheduler;
2
3 import org.springframework.boot.SpringApplication;
4
5
6 @SpringBootApplication
7 public class MeetingSchedulerApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(MeetingSchedulerApplication.class, args);
11     }
12 }
13
14 }
```

MeetingSchedulerException.java (Exception Class) -> Already Implemented

```
MeetingSchedulerException.java X
1 package com.infy.meetingscheduler.exception;
2
3 // Already Implemented
4 public class MeetingSchedulerException extends Exception{
5
6
7     private static final long serialVersionUID = 1L;
8
9     public MeetingSchedulerException(String message) {
10         super(message);
11     }
12
13 }
```

Application.properties (Properties file) -> Already implemented

```
application.properties X
1 MeetingValidator.INVALID_TEAM_NAME=Team Name should be one among ETAMYSJAVA, ETAMYSMS, ETAMYSBI, ETAMYSUI, ETAMYSAI
2 MeetingValidator.INVALID_MEETING_DATE=Meeting Date cannot be a Saturday or a Sunday
3 MeetingService.NO_MEETINGS_FOUND=No Meetings were found to the given Scheduler
4 MeetingService.MEETING_DATE_UNAVAILABLE=The scheduler has a meeting on the requested date, please choose any other date
5 MeetingService.TEAM_UNAVAILABLE=The team has a meeting on the requested date, please choose any other date
6 General.EXCEPTION_MESSAGE=Internal Server Error!
7
8 # Database properties
9 spring.datasource.url=jdbc:mysql://localhost:3306/infymeeting_db
10 spring.datasource.username=root
11 spring.datasource.password=root
12 spring.jpa.show-sql=true
13 spring.jpa.properties.hibernate.format_sql=true
14
15 server.port=8765
16
```

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

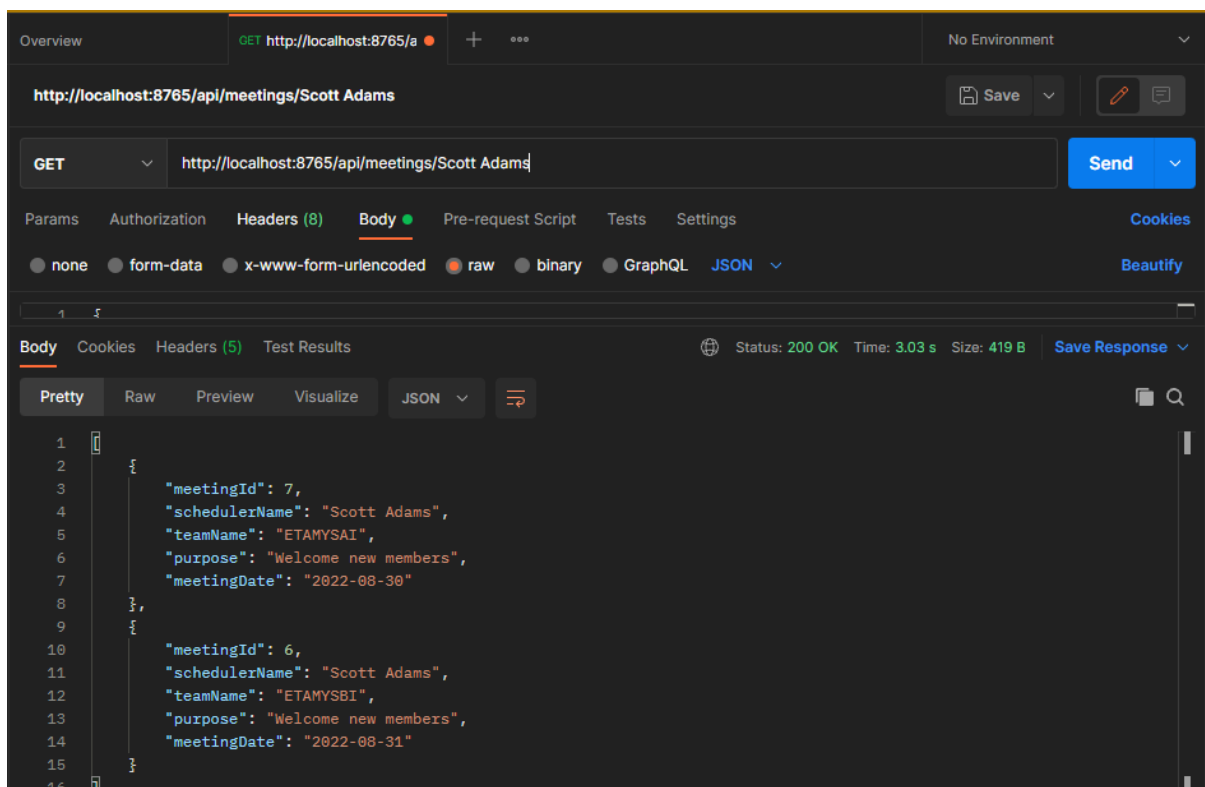
```
ValidationMessages.properties X
1 meeting.scheduler.invalid=Scheduler name should have only letters and spaces
2 meeting.scheduler.absent=Scheduler name is mandatory
3 meeting.team.absent=Team name is mandatory
4 meeting.purpose.absent=Purpose is mandatory
5 meeting.date.absent=Meeting date is mandatory
6 meeting.date.invalid=Meeting date should not be a past date
```


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

```
log4j2.properties X
1 name=LoggingFile
2 rootLogger.level=info
3 rootLogger.appenderRef.file.ref=LoggerAppender
4
5 appender.file.type=File
6 appender.file.name=LoggerAppender
7 appender.file.fileName=log/ErrorLog.log
8 appender.file.layout.type=PatternLayout
9 appender.file.layout.pattern=%d{dd-MMM-yyyy HH:mm:ss} %level - %m%n
10
11 # Console Appender
12 appender.console.name=ConsoleAppender
13 appender.console.type=Console
14 appender.console.layout.type=PatternLayout
15 appender.console.layout.pattern=%m%n
16
17
18 #Declaring logger for business logic
19 logger.infyacademy.name=com.infy.meetingscheduler.utility
20 logger.infyacademy.level=DEBUG
21 logger.infyacademy.appenderRef.file.ref=LoggerAppender
22 logger.infyacademy.additivity=false
23
24 logger.testster.name=com.infy.meetingscheduler
25 logger.testster.level=INFO
26 logger.testster.appenderRef.file.ref=ConsoleAppender
27 logger.testster.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

The screenshot shows the Postman interface for a GET request to `http://localhost:8765/api/meetings/123`. The request is saved and the response is displayed in the 'Body' tab. The status is **400 Bad Request** with a time of 2.24 s and size of 229 B. The response body is formatted as JSON and shows an error message: `"errorMessage": "Scheduler name should have only letters and spaces",` and `"errorCode": 400`.

```
{  "errorMessage": "Scheduler name should have only letters and spaces",  "errorCode": 400}
```

3. GET: Invalid Test Data 2

The screenshot shows the Postman interface for a GET request to `http://localhost:8765/api/meetings/xyz`. The request is saved and the response is displayed in the 'Body' tab. The status is **400 Bad Request** with a time of 2.56 s and size of 224 B. The response body is formatted as JSON and shows an error message: `"errorMessage": "No Meetings were found to the given Scheduler",` and `"errorCode": 400`.

```
{  "errorMessage": "No Meetings were found to the given Scheduler",  "errorCode": 400}
```

4. POST: Valid Test Data

The screenshot shows the Postman interface for a POST request to `http://localhost:8765/api/meetings`. The request body is a valid JSON object:

```
1 {
2   "schedulerName": "Jamie Spencer",
3   "teamName": "ETAMYSJAVA",
4   "purpose": "Welcome new members",
5   "meetingDate": "2022-09-09"
6 }
```

The response is displayed in the "Body" tab, showing a successful status of 201 Created:

```
1 {
2   "meetingId": 8,
3   "schedulerName": "Jamie Spencer",
4   "teamName": "ETAMYSJAVA",
5   "purpose": "Welcome new members",
6   "meetingDate": "2022-09-09"
7 }
```

Additional details: Status: 201 Created, Time: 5.04 s, Size: 299 B.

5. POST: Invalid Test Data 1

The screenshot shows the Postman interface for a POST request to `http://localhost:8765/api/meetings`. The request body is invalid JSON:

```
1 {
2   "schedulerName": "Jamie123",
3   "teamName": "ETAMYSJAVA",
4   "meetingDate": "2022-09-14"
5 }
```

The response is displayed in the "Body" tab, showing a 400 Bad Request status:

```
1 {
2   "errorMessage": "Purpose is mandatory, Scheduler name should have only letters and spaces",
3   "errorCode": 400
4 }
```

Additional details: Status: 400 Bad Request, Time: 155 ms, Size: 251 B.

6. POST: Invalid Test Data 2

The screenshot shows a Postman interface for a POST request to `http://localhost:8765/api/meetings`. The request body is a JSON object:

```
{  "schedulerName": "Jamie",  "teamName": "ETAMYS",  "purpose": "Welcome new members",  "meetingDate": "2022-09-14"}
```

The response is a 400 Bad Request with the following JSON body:

```
{  "errorMessage": "Team Name should be one among ETAMYSJAVA, ETAMYSMS, ETAMYSBI, ETAMYSUI, ETAMYSAI",  "errorCode": 400}
```

Response details: Status: 400 Bad Request, Time: 104 ms, Size: 259 B.

7. POST: Invalid Test Data 3

The screenshot shows a Postman interface for a POST request to `http://localhost:8765/api/meetings`. The request body is a JSON object:

```
{  "schedulerName": "Jamie",  "teamName": "ETAMYSJAVA",  "purpose": "Welcome new members",  "meetingDate": "2022-09-03"}
```

The response is a 400 Bad Request with the following JSON body:

```
{  "errorMessage": "Meeting Date cannot be a Saturday or a Sunday",  "errorCode": 400}
```

Response details: Status: 400 Bad Request, Time: 264 ms, Size: 224 B.

8. POST: Invalid Test Data 4

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

- Method:** POST
- URL:** http://localhost:8765/api/meetings
- Body (JSON):**

```
{  "schedulerName": "Scott Adams",  "teamName": "ETAMYSBI",  "purpose": "Welcome new members",  "meetingDate": "2022-08-30"}
```
- Status:** 400 Bad Request
- Time:** 64 ms
- Size:** 258 B
- Response Body (JSON):**

```
{  "errorMessage": "The scheduler has a meeting on the requested date, please choose any other date",  "errorCode": 400}
```

9. POST: Invalid Test Data 5

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

- Method:** POST
- URL:** http://localhost:8765/api/meetings
- Body (JSON):**

```
{  "schedulerName": "Mathew White",  "teamName": "ETAMYSAI",  "purpose": "Welcome new members",  "meetingDate": "2022-08-30"}
```
- Status:** 400 Bad Request
- Time:** 153 ms
- Size:** 253 B
- Response Body (JSON):**

```
{  "errorMessage": "The team has a meeting on the requested date, please choose any other date",  "errorCode": 400}
```