Solution:

Once the application run it fetches the latest employee meeting data using the API and saves it in a local text file. Then it takes inputs for meeting slots i.e. employee ID/IDs, desired meeting times slots with dates, meeting length in minutes, office start and end times. Then it creates two lists, one for the already booked meetings of desired employee/employees on the desired date-time and another one for possible time slots for desired employee/employees on the desired date-time. Then using both the lists we extract the meeting slots in which employee/employees are available. In the end, this list is filtered out, based on the office timings to show the meeting available between office hours.

Code Flow:

officeHourFilter();
displayAvailableSlots();

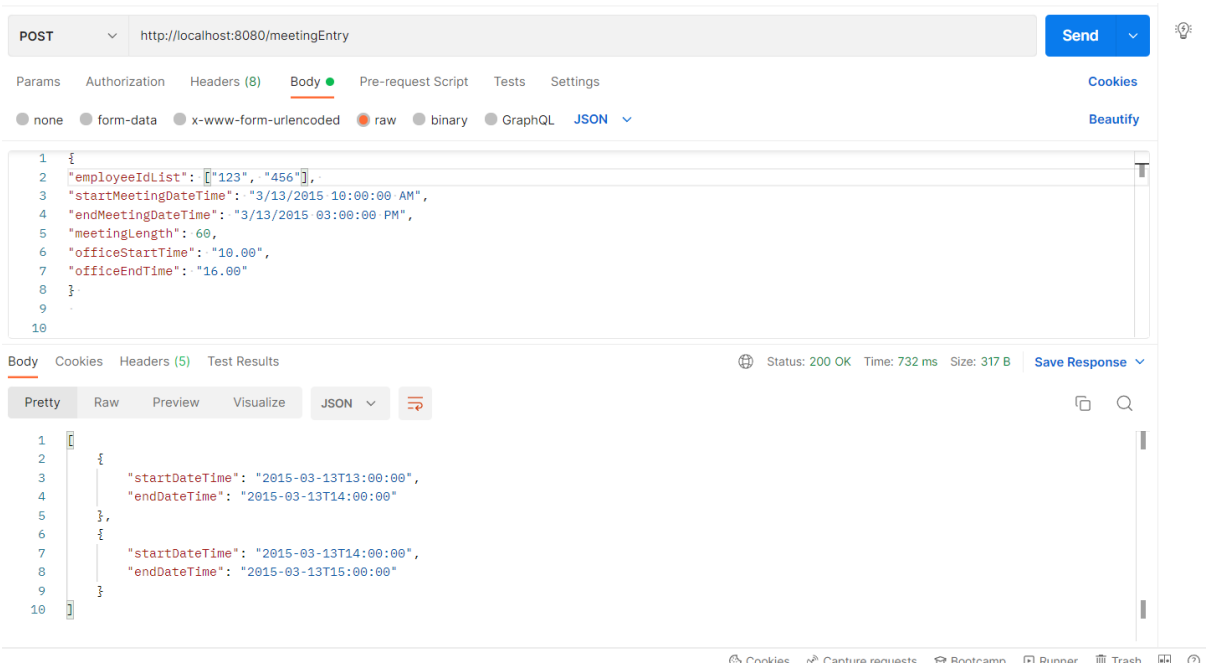
- Possible time slots list filter out with the office timing and prepare final available slots for a meeting.
- Display the final available slots list

How to run the application:

Please follow the following step to run the application.
1. Clone the below-mentioned project git repository at the local system
https://github.com/Saurabh311/assignment-Lime-technology.git

2. Open the project in IDE and run the file name “AssignmentLimeTechnologyApplication”. Path for this file: assignmentLimeTechnology -> src -> main -> java -> com.example.assignmentlimetechnology -> AssignmentLimeTechnologyApplication

3. After running the code, open the postman from where you can send the request to get the response with available time slot if its time slots available otherwise it responds with blank array if no slot are available. In console you will get the response with message as well. Screen shot is attached for your reference.



4. For unit test cases, run the file AssignmentLimeTechnologyApplicationTest by following this path: assignmentLimeTechnology -> src -> test -> java -> com.example.assignmentlimetechnology -> AssignmentLimeTechnologyApplicationTests.

5. You can check the test case results by running the file AssignmentLimeTechnologyApplicationTest.

6. Data using API is saved in resource folder under main folder named “bookedMeetingFile.txt” and hardcode “testcase.txt” file also saved for testing purpose.

7. Test Case scenarios with examples are mentioned below for reference.

Case Scenarios:

Case-1
When all employees are busy in desired time frame.

Input:(using Postman)
Post: http://localhost:8080/meetingEntry


```
{
"employeeIdList": ["123", "456", "789"],
"startMeetingDateTime": "3/13/2015 10:00:00 AM",
"endMeetingDateTime": "3/13/2015 03:00:00 PM",
"meetingLength": 60,
"officeStartTime": "10.00",
"officeEndTime": "16.00"
}
```


In this case scenario we are passing ids of three employees and looking for available time slots for the date of 03/13/2015 between 10AM-15PM of 60 minutes. Result will be “No time slots available for meeting” because as per saved data slots are already booked for desired time. It will respond blank array on postman.

Case-2
When some slots are available in desired time slot.

Input:(using Postman)
Post: http://localhost:8080/meetingEntry


```
{
"employeeIdList": ["123","456"],
"startMeetingDateTime": "3/14/2015 08:00:00 AM",
"endMeetingDateTime": "3/14/2015 12:00:00 PM",
"meetingLength": 60,
"officeStartTime": "08.00",
"officeEndTime": "16.00"
}
```


In this case scenario we are passing ids of two employees and looking for available time slots for the date of 03/14/2015 between 08AM-12PM of 60 minutes. Result will be one meeting slot available between 08-09 AM.

Case-3
When two employees are busy on same time

Input:(using Postman)
Post: http://localhost:8080/meetingEntry

```
{
"employeeIdList": ["123","456"],
"startMeetingDateTime": "3/15/2015 10:00:00 AM",
"endMeetingDateTime": "3/15/2015 01:00:00 PM",
"meetingLength": 60,
"officeStartTime": "09.00",
"officeEndTime": "14.00"
}
```

In this case scenario we are passing ids of two employees and looking for available time slots for the date of 03/15/2015 between 10AM - 01PM of 60 minutes. There will be two meeting slot available between 10AM -11AM and 11:00AM – 12:00PM

Case - 04
When user want to know the meetings slot between two days.

Input:(using Postman)
Post: http://localhost:8080/meetingEntry

```
{
"employeeIdList": ["123","456"],
"startMeetingDateTime": "3/16/2015 9:00:00 AM",
"endMeetingDateTime": "3/17/2015 04:00:00 PM",
"meetingLength": 30,
"officeStartTime": "08.00",
"officeEndTime": "17.00"
}
```

In this case scenario we are passing ids of two employees and looking for available time slots for 2 days from 03/16/2015 to 03/17/2015 between 9AM - 4PM of 30 minutes. There will be few slots available on both the dates.

Case – 05
Wrong Id input

Input:(using Postman)
Post: http://localhost:8080/meetingEntry

```
{
"employeeIdList": ["123","456wrongID", "789"],
"startMeetingDateTime": "3/16/2015 9:00:00 AM",
"endMeetingDateTime": "3/17/2015 04:00:00 PM",
"meetingLength": 30,
"officeStartTime": "08.00",
"officeEndTime": "17.00"
}
```

In this case scenario we pass one wrong ID. It will throw an exception with wrong Id on console. Response will be status -500 on postman.

Case – 06
When desired meeting length is 1:30 hour.

Input:(using Postman)
Post: http://localhost:8080/meetingEntry


```
{
"employeeIdList": ["123","456", "789"],
"startMeetingDateTime": "3/16/2015 10:00:00 AM",
"endMeetingDateTime": "3/17/2015 02:00:00 PM",
"meetingLength": 90,
"officeStartTime": "08.00",
"officeEndTime": "17.00"
}
```


In this case scenario result, be shown as per 1:30 hour meeting length.