**CAPSTONE I (CS 392)**

**LABORATORY MANUAL**

**B.Tech. (CSE/ DS/CYS/iMSC)**

**Semester II**

**L-T-P-C: 2-0-4-4**

**Academic Year: 2024-25**

**A logo of a university

Description automatically generated**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

|  |  |
| --- | --- |
| Student ID | BT22GCS108 |
| Name | ABHI BHARDWAJ |

**NIIT UNIVERSITY, NEEMRANA**

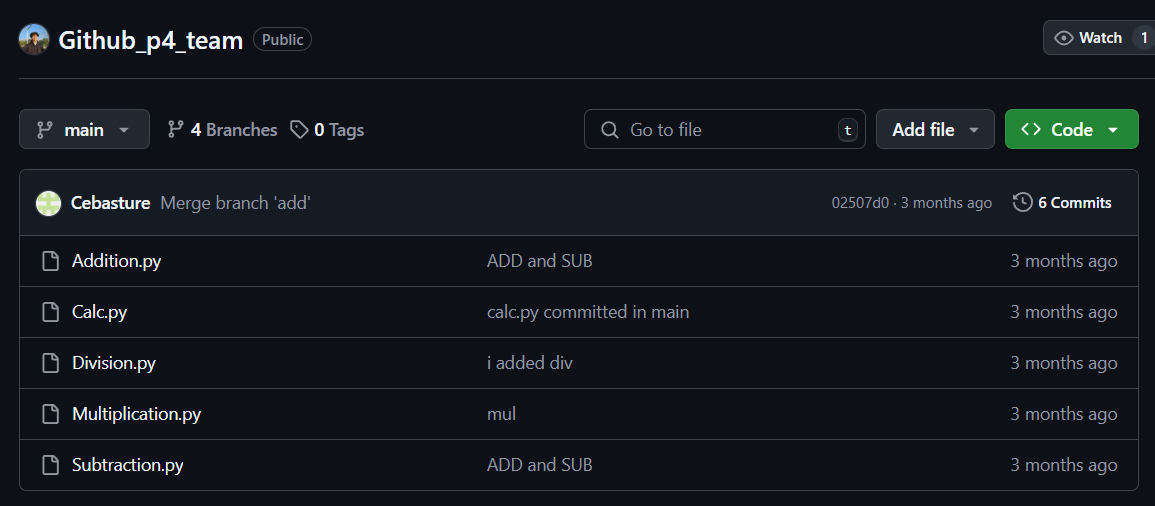
NH-8, Delhi-Jaipur Highway,  
Neemrana, District Alwar (Rajasthan),  
Pin-301705  
Website: [www.niituniversity.in](http://www.niituniversity.in)

**List of Experiments**

|  |  |  |  |
| --- | --- | --- | --- |
| **Practical No.** | **Aim of Practical** | **COs** | **Page No.** |
| 1 | Collaboration Tool-Git | CO2 | 10 |
| 2 | HTML and CSS | CO1 | 10 |
| 3 | Javascript | CO1 | 10 |
| 4 | BootStrap | CO1 | 19 |
| 5 | NodeJS | CO3 | 19 |
| 6 | ExpressJS | CO3 | 19 |
| 7 | MongoDB | CO3 | 20 |
| 8 | Database Connectivity | CO3 | 27 |
| 9 | Login Exercise | CO3 | 27 |
| 10 | Session and Cookies Demo | CO3 | 28 |
| 11 | Setting up DevOPS for Project | CO4  CO5 | 28 |
| 12 | APIs | CO4  CO5 | 28 |

**Practical 1:** Collaboration Tool-Git:

**Steps of Execution:**

1. Team lead will make Github repository and will send invite to other team members.
2. Team lead will create local repo and then push to remote repo
3. Team members will accept the invite and clone repo on their local machine.
4. All the team members will share file using git / github
5. Team members will update their repository and check the contents shared by others.



**Practical 2:** HTML and CSS:

**Steps of Execution:**

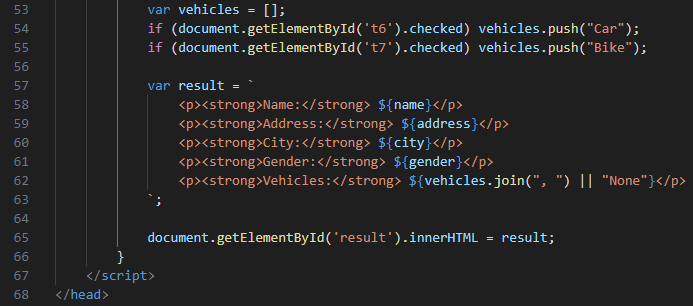
1. Make an input form for an employee.
2. Apply the relevant CSS using external CSS file

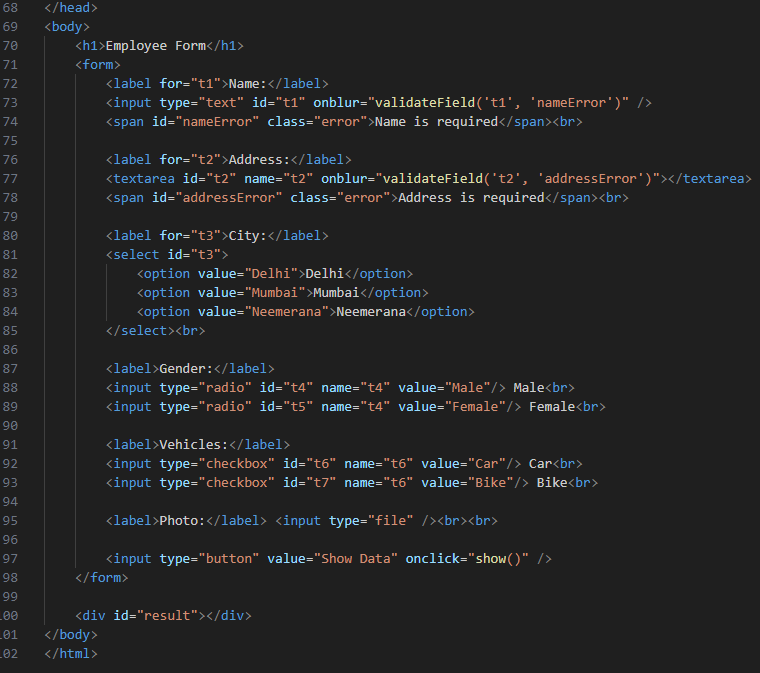
**Input Html file :**





**Html file continues :**

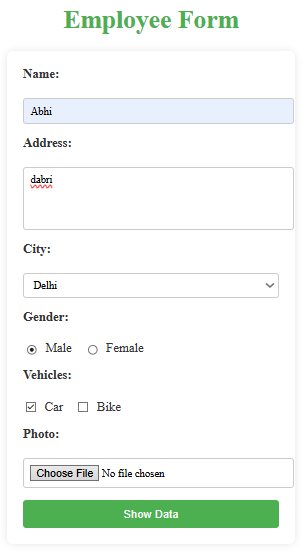


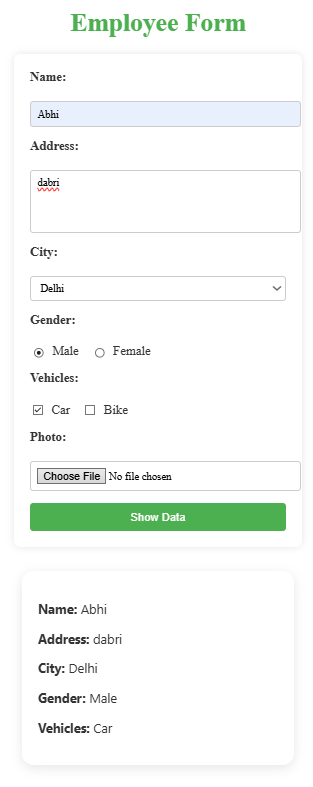


**Input CSS file :**



**Output :**



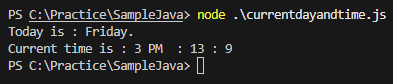
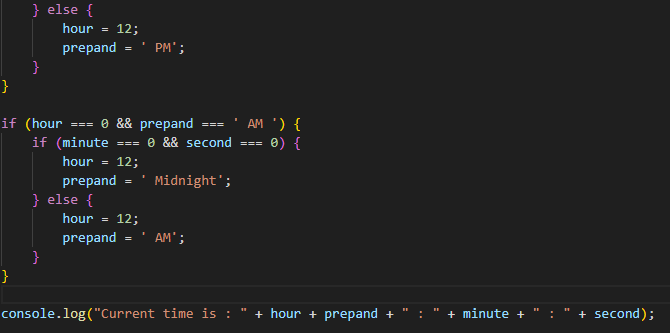


**Practical 3:** Javascript:

Write a JavaScript program to display the current day and time in the following format.  
Today is : Tuesday.   
Current time is : 10 PM : 30 : 38

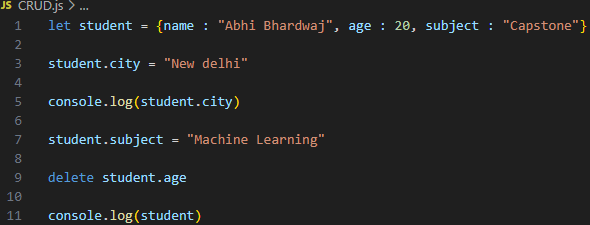
**Input Code :**

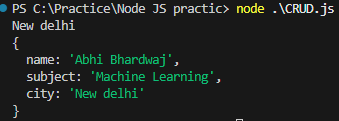


 **Output:   
  
  
  
  
Practical 4:** BootStrap :

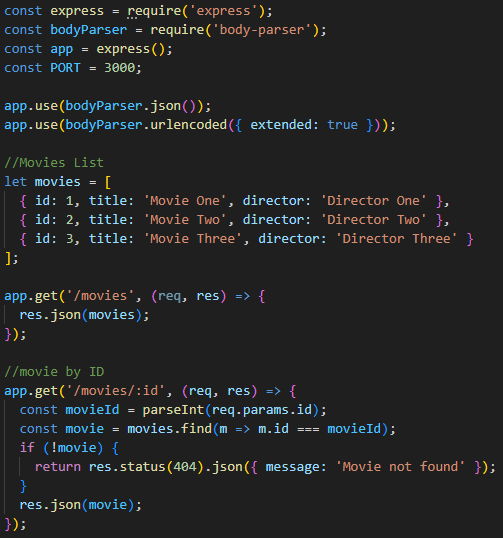
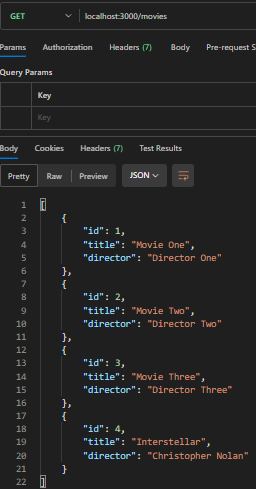
<https://github.com/4GeeksAcademy/bootstrap-exercises-tutorial>

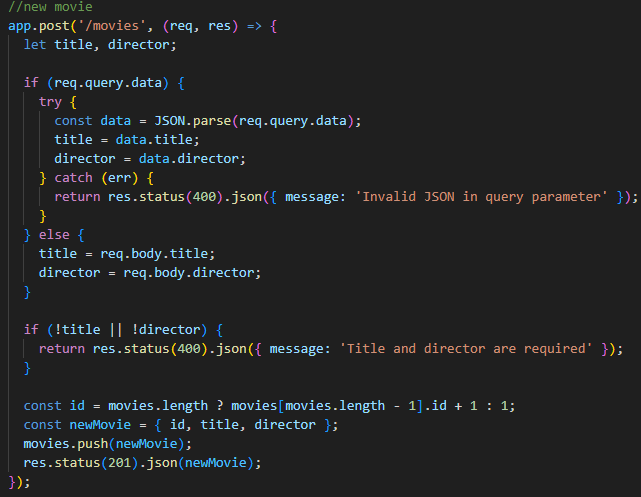
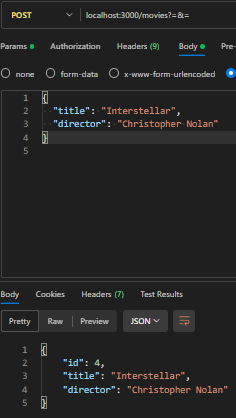
**Practical 5:** NodeJS :

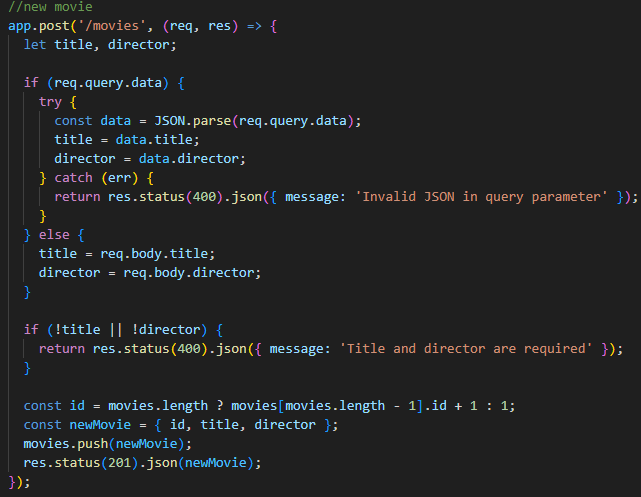
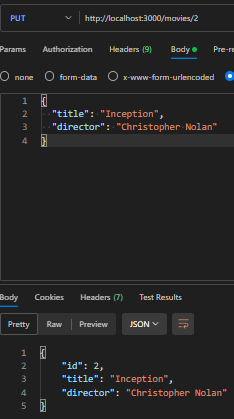
Do CRUD operation using NodeJS with MySQL

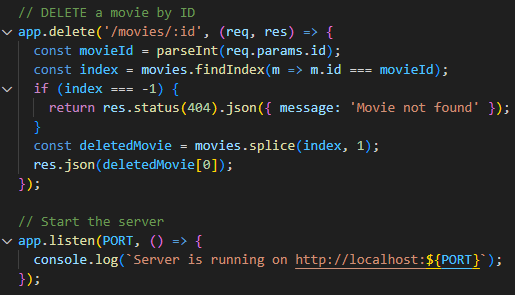
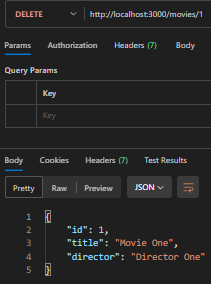
 **Practical 6:** ExpressJS :

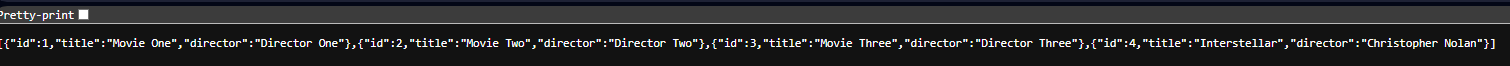
Make use of GET, POST, PUT and DELETE HTTP methods with ExpressJS

**Input code Postman window**







**Output Windows**  
  
**After PUT :**  
  
  
  
  
  
1. Write a MongoDB query to display all the documents in the collection restaurants.   
**Ans** : db.restaurants.find()

2. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine for all the documents in the collection restaurant.   
**Ans** : db.restaurants.find({}, { restaurant\_id: 1, name: 1, borough: 1, cuisine: 1 })

3. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine, but exclude the field \_id for all the documents in the collection restaurant.   
**Ans** : db.restaurants.find({}, { \_id: 0, restaurant\_id: 1, name: 1, borough: 1, cuisine: 1 })

4. Write a MongoDB query to display the fields restaurant\_id, name, borough and zip code, but exclude the field \_id for all the documents in the collection restaurant.   
 **Ans** : db.restaurants.find({}, { \_id: 0, restaurant\_id: 1, name: 1, borough: 1, "address.zipcode": 1 })

5. Write a MongoDB query to display all the restaurant which is in the borough Bronx.   
**Ans** : db.restaurants.find({ borough: "Bronx" })

6. Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx.   
**Ans** : db.restaurants.find({ borough: "Bronx" }).limit(5)

7.Write a MongoDB query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx.   
**Ans** : db.restaurants.find({ borough: "Bronx" }).skip(5).limit(5)

8. Write a MongoDB query to find the restaurants who achieved a score more than 90.   
**Ans** : db.restaurants.find({ "grades.score": { $gt: 90 } })

9. Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100.   
**Ans** : db.restaurants.find({ "grades.score": { $gt: 80, $lt: 100 } })

10. Write a MongoDB query to find the restaurants which locate in latitude value less than -95.754168.  
**Ans** : db.restaurants.find({ "address.coord.1": { $lt: -95.754168 } })

11. Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168.   
**Ans** : db.restaurants.find({

cuisine: { $ne: "American" },

"grades.score": { $gt: 70 },

"address.coord.1": { $lt: -65.754168 }

})

12. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168.  
Note : Do this query without using $and operator.   
**Ans** : db.restaurants.find({

cuisine: { $ne: "American" },

"grades.score": { $gt: 70 },

"address.coord.1": { $lt: -65.754168 }

})

13. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a grade point 'A' not belongs to the borough Brooklyn. The document must be displayed according to the cuisine in descending order.   
**Ans** : db.restaurants.find({

cuisine: { $ne: "American" },

"grades.grade": "A",

borough: { $ne: "Brooklyn" }

}).sort({ cuisine: -1 })

14. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.   
**Ans** : db.restaurants.find(

{ name: /^Wil/ },

{ restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 }

)

15. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.   
**Ans** : db.restaurants.find(

{ name: /ces$/ },

{ restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 }

)

16. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name.   
**Ans** : db.restaurants.find(

{ name: /Reg/ },

{ restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 }

)

17. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish.   
**Ans** : db.restaurants.find({

borough: "Bronx",

cuisine: { $in: ["American", "Chinese"] }

})

18. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronxor Brooklyn.   
**Ans** : db.restaurants.find(

{ borough: { $in: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } },

{ restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 }

)

19. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island or Queens or Bronxor Brooklyn.   
**Ans** : db.restaurants.find(

{ borough: { $nin: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } },

{ restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 }

)

20. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which achieved a score which is not more than 10.   
**Ans** : db.restaurants.find(

{ "grades.score": { $lte: 10 } },

{ restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 }

)

21. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'.   
**Ans** : db.restaurants.find(

{

$or: [

{ name: /^Wil/ },

{

$and: [

{ cuisine: { $ne: "American" } },

{ cuisine: { $ne: "Chinese" } }

]

}

]

},

{ restaurant\_id:

::contentReference[oaicite:23]{index=23}

22. Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many of survey dates..   
**Ans** : db.restaurants.find({

grades: {

$elemMatch: {

grade: "A",

score: 11,

date: ISODate("2014-08-11T00:00:00Z")

}

}

}, {

restaurant\_id: 1,

name: 1,

grades: 1,

\_id: 0

})

23. Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z".   
**Ans** : db.restaurants.find({

"grades.1.grade": "A",

"grades.1.score": 9,

"grades.1.date": ISODate("2014-08-11T00:00:00Z")

}, {

restaurant\_id: 1,

name: 1,

grades: 1,

\_id: 0

})

24. Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52..   
**Ans** : db.restaurants.find({

"address.coord.1": { $gt: 42, $lte: 52 }

}, {

restaurant\_id: 1,

name: 1,

address: 1,

"address.coord": 1,

\_id: 0

})

25. Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns.   
**Ans** : db.restaurants.find().sort({ name: 1 })

26. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.   
**Ans** : db.restaurants.find().sort({ name: -1 })

27. Write a MongoDB query to arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.   
**Ans** : db.restaurants.find({}, {

name: 1,

borough: 1,

cuisine: 1,

\_id: 0

}).sort({ cuisine: 1, borough: -1 })

28. Write a MongoDB query to know whether all the addresses contains the street or not.   
**Ans** : db.restaurants.find({ "address.street": { $exists: false } })

29. Write a MongoDB query which will select all documents in the restaurants collection where the coord field value is Double.   
**Ans** : db.restaurants.find({

"address.coord.0": { $type: "double" },

"address.coord.1": { $type: "double" }

})

30. Write a MongoDB query which will select the restaurant Id, name and grades for those restaurants which returns 0 as a remainder after dividing the score by 7.   
**Ans** : db.restaurants.find({

"grades.score": { $mod: [7, 0] }

}, {

restaurant\_id: 1,

name: 1,

grades: 1,

\_id: 0

})

31. Write a MongoDB query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name.   
**Ans** : db.restaurants.find({

name: /mon/i

}, {

name: 1,

borough: 1,

longitude: "$address.coord.0",

latitude: "$address.coord.1",

cuisine: 1,

\_id: 0

})

32. Write a MongoDB query to find the restaurant name, borough, longitude and latitude and cuisine for those restaurants which contain 'Mad' as first three letters of its name.   
**Ans** : db.restaurants.find({

name: /^Mad/

}, {

name: 1,

borough: 1,

longitude: "$address.coord.0",

latitude: "$address.coord.1",

cuisine: 1,

\_id: 0

})

33. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5.   
**Ans** : db.restaurants.find({

"grades.score": { $lt: 5 }

})

34. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan.   
**Ans** : db.restaurants.find({

borough: "Manhattan",

"grades.score": { $lt: 5 }

})

35. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn.   
**Ans** : db.restaurants.find({

borough: { $in: ["Manhattan", "Brooklyn"] },

"grades.score": { $lt: 5 }

})

36. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.   
**Ans** : db.restaurants.find({

borough: { $in: ["Manhattan", "Brooklyn"] },

cuisine: { $ne: "American" },

"grades.score": { $lt: 5 }

})

37. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.   
**Ans** : db.restaurants.find({

borough: { $in: ["Manhattan", "Brooklyn"] },

cuisine: { $nin: ["American", "Chinese"] },

"grades.score": { $lt: 5 }

})

38. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6.   
**Ans** : db.restaurants.find({

"grades.score": 2,

"grades.score": 6

})

39. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan.   
**Ans** : db.restaurants.find({

borough: "Manhattan",

"grades.score": 2,

"grades.score": 6

})

40. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn.   
**Ans** : db.restaurants.find({

borough: { $in: ["Manhattan", "Brooklyn"] },

"grades.score": 2,

"grades.score": 6

})

41. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.   
**Ans** : db.restaurants.find({

borough: { $in: ["Manhattan", "Brooklyn"] },

cuisine: { $ne: "American" },

"grades.score": 2,

"grades.score": 6

})

42. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.   
**Ans** : db.restaurants.find({

grades: { $all: [{ $elemMatch: { score: 2 } }, { $elemMatch: { score: 6 } }] },

borough: { $in: ["Manhattan", "Brooklyn"] },

cuisine: { $nin: ["American", "Chinese"] }

})

43. Write a MongoDB query to find the restaurants that have a grade with a score of 2 or a grade with a score of 6.   
**Ans** : db.restaurants.find({

"grades.score": { $in: [2, 6] }

})

44. Write a MongoDB query to find the restaurants that have a grade with a score of 2 or a grade with a score of 6 and are located in the borough of Manhattan.   
**Ans** : db.restaurants.find({

borough: "Manhattan",

"grades.score": { $in: [2, 6] }

})

45. Write a MongoDB query to find the restaurants that have a grade with a score of 2 or a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn.   
**Ans** : db.restaurants.find({

borough: { $in: ["Manhattan", "Brooklyn"] },

"grades.score": { $in: [2, 6] }

})

46. Write a MongoDB query to find the restaurants that have a grade with a score of 2 or a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.   
**Ans** : db.restaurants.find({

borough: { $in: ["Manhattan", "Brooklyn"] },

cuisine: { $ne: "American" },

"grades.score": { $in: [2, 6] }

})

47. Write a MongoDB query to find the restaurants that have a grade with a score of 2 or a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.   
**Ans** : db.restaurants.find({

borough: { $in: ["Manhattan", "Brooklyn"] },

cuisine: { $nin: ["American", "Chinese"] },

"grades.score": { $in: [2, 6] }

})

48. Write a MongoDB query to find the restaurants that have all grades with a score greater than 5.   
**Ans** : db.restaurants.find({

grades: { $not: { $elemMatch: { score: { $lte: 5 } } } }

})

49. Write a MongoDB query to find the restaurants that have all grades with a score greater than 5 and are located in the borough of Manhattan.   
**Ans** : db.restaurants.find({

borough: "Manhattan",

grades: { $not: { $elemMatch: { score: { $lte: 5 } } } }

})

50. Write a MongoDB query to find the restaurants that have all grades with a score greater than 5 and are located in the borough of Manhattan or Brooklyn.   
**Ans** : db.restaurants.find({

borough: { $in: ["Manhattan", "Brooklyn"] },

grades: { $not: { $elemMatch: { score: { $lte: 5 } } } }

})

51. Write a MongoDB query to find the average score for each restaurant.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $group: {

\_id: "$restaurant\_id",

name: { $first: "$name" },

avgScore: { $avg: "$grades.score" }

}}

])

52. Write a MongoDB query to find the highest score for each restaurant.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $group: {

\_id: "$restaurant\_id",

name: { $first: "$name" },

maxScore: { $max: "$grades.score" }

}}

])

53. Write a MongoDB query to find the lowest score for each restaurant.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $group: {

\_id: "$restaurant\_id",

name: { $first: "$name" },

minScore: { $min: "$grades.score" }

}}

])

54. Write a MongoDB query to find the count of restaurants in each borough.   
**Ans** : db.restaurants.aggregate([

{ $group: {

\_id: "$borough",

count: { $sum: 1 }

}}

])

55. Write a MongoDB query to find the count of restaurants for each cuisine.   
**Ans** : db.restaurants.aggregate([

{ $group: {

\_id: "$cuisine",

count: { $sum: 1 }

}}

])

56. Write a MongoDB query to find the count of restaurants for each cuisine and borough.   
**Ans** : db.restaurants.aggregate([

{ $group: {

\_id: { cuisine: "$cuisine", borough: "$borough" },

count: { $sum: 1 }

}}

])

57. Write a MongoDB query to find the count of restaurants that received a grade of 'A' for each cuisine.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $match: { "grades.grade": "A" } },

{ $group: {

\_id: "$cuisine",

count: { $sum: 1 }

}}

])

58. Write a MongoDB query to find the count of restaurants that received a grade of 'A' for each borough.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $match: { "grades.grade": "A" } },

{ $group: {

\_id: "$borough",

count: { $sum: 1 }

}}

])

59. Write a MongoDB query to find the count of restaurants that received a grade of 'A' for each cuisine and borough.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $match: { "grades.grade": "A" } },

{ $group: {

\_id: { cuisine: "$cuisine", borough: "$borough" },

count: { $sum: 1 }

}}

])

60. Write a MongoDB query to find the number of restaurants that have been graded in each month of the year.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $project: {

month: { $month: "$grades.date" }

}},

{ $group: {

\_id: "$month",

count: { $sum: 1 }

}}

])

61. Write a MongoDB query to find the average score for each cuisine.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $group: {

\_id: "$cuisine",

avgScore: { $avg: "$grades.score" }

}}

])

62. Write a MongoDB query to find the highest score for each cuisine.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $group: {

\_id: "$cuisine",

maxScore: { $max: "$grades.score" }

}}

])

63. Write a MongoDB query to find the lowest score for each cuisine.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $group: {

\_id: "$cuisine",

minScore: { $min: "$grades.score" }

}}

])

64. Write a MongoDB query to find the average score for each borough.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $group: {

\_id: "$borough",

avgScore: { $avg: "$grades.score" }

}}

])

65. Write a MongoDB query to find the highest score for each borough.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $group: {

\_id: "$borough",

maxScore: { $max: "$grades.score" }

}}

])

66. Write a MongoDB query to find the lowest score for each borough.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $group: {

\_id: "$borough",

minScore: { $min: "$grades.score" }

}}

])

67. Write a MongoDB query to find the name and address of the restaurants that received a grade of 'A' on a specific date.   
**Ans** : db.restaurants.find({

grades: {

$elemMatch: {

grade: "A",

date: ISODate("2014-08-11T00:00:00Z")

}

}

}, {

name: 1,

address: 1,

\_id: 0

});

68. Write a MongoDB query to find the name and address of the restaurants that received a grade of 'B' or 'C' on a specific date.   
**Ans** : db.restaurants.find({

grades: {

$elemMatch: {

grade: { $in: ["B", "C"] },

date: ISODate("2014-08-11T00:00:00Z")

}

}

}, {

name: 1,

address: 1,

\_id: 0

});

69. Write a MongoDB query to find the name and address of the restaurants that have at least one 'A' grade and one 'B' grade.   
**Ans** : db.restaurants.find({

$and: [

{ "grades.grade": "A" },

{ "grades.grade": "B" }

]

}, {

name: 1,

address: 1,

\_id: 0

});

70. Write a MongoDB query to find the name and address of the restaurants that have at least one 'A' grade and no 'B' grades.   
**Ans** : db.restaurants.find({

"grades.grade": "A",

"grades.grade": { $ne: "B" }

}, {

name: 1,

address: 1,

\_id: 0

});

71. Write a MongoDB query to find the name ,address and grades of the restaurants that have at least one 'A' grade and no 'C' grades.   
**Ans** : db.restaurants.find({

"grades.grade": "A",

"grades.grade": { $ne: "C" }

}, {

name: 1,

address: 1,

grades: 1,

\_id: 0

});

72. Write a MongoDB query to find the name, address, and grades of the restaurants that have at least one 'A' grade, no 'B' grades, and no 'C' grades.   
**Ans** : db.restaurants.find({

"grades.grade": "A",

"grades.grade": { $nin: ["B", "C"] }

}, {

name: 1,

address: 1,

grades: 1,

\_id: 0

});

73. Write a MongoDB query to find the name and address of the restaurants that have the word 'coffee' in their name.   
**Ans** : db.restaurants.find({

name: /coffee/i

}, {

name: 1,

address: 1,

\_id: 0

});

74. Write a MongoDB query to find the name and address of the restaurants that have a zipcode that starts with '10'.   
**Ans** : db.restaurants.find({

"address.zipcode": /^10/

}, {

name: 1,

address: 1,

\_id: 0

});

75. Write a MongoDB query to find the name and address of the restaurants that have a cuisine that starts with the letter 'B'.   
**Ans** : db.restaurants.find({

cuisine: /^B/

}, {

name: 1,

address: 1,

\_id: 0

});

76. Write a MongoDB query to find the name, address, and cuisine of the restaurants that have a cuisine that ends with the letter 'y'.   
**Ans** : db.restaurants.find({

cuisine: /y$/

}, {

name: 1,

address: 1,

cuisine: 1,

\_id: 0

});

77. Write a MongoDB query to find the name, address, and cuisine of the restaurants that have a cuisine that contains the word 'Pizza'.   
**Ans** : db.restaurants.find({

cuisine: /Pizza/i

}, {

name: 1,

address: 1,

cuisine: 1,

\_id: 0

});

78. Write a MongoDB query to find the restaurants achieved highest average score.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $group: { \_id: "$name", avgScore: { $avg: "$grades.score" } } },

{ $sort: { avgScore: -1 } },

{ $limit: 1 }

]);

79. Write a MongoDB query to find all the restaurants with the highest number of "A" grades.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $match: { "grades.grade": "A" } },

{ $group: { \_id: "$name", aCount: { $sum: 1 } } },

{ $sort: { aCount: -1 } },

{ $limit: 1 }

]);

80. Write a MongoDB query to find the cuisine type that is most likely to receive a "C" grade.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $match: { "grades.grade": "C" } },

{ $group: { \_id: "$cuisine", count: { $sum: 1 } } },

{ $sort: { count: -1 } },

{ $limit: 1 }

]);

81. Write a MongoDB query to find the restaurant that has the highest average score for thecuisine "Turkish".   
**Ans** : db.restaurants.aggregate([

{ $match: { cuisine: "Turkish" } },

{ $unwind: "$grades" },

{ $group: { \_id: "$name", avgScore: { $avg: "$grades.score" } } },

{ $sort: { avgScore: -1 } },

{ $limit: 1 }

]);

82. Write a MongoDB query to find the restaurants that achieved the highest total score.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $group: { \_id: "$name", totalScore: { $sum: "$grades.score" } } },

{ $sort: { totalScore: -1 } },

{ $limit: 1 }

]);

83. Write a MongoDB query to find all the Chinese restaurants in Brooklyn.   
**Ans** : db.restaurants.find({

borough: "Brooklyn",

cuisine: "Chinese"

}, {

name: 1,

address: 1,

\_id: 0

});

84. Write a MongoDB query to find the restaurant with the most recent grade date.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $sort: { "grades.date": -1 } },

{ $limit: 1 },

{ $project: { name: 1, address: 1, recentGrade: "$grades" } }

]);

85. Write a MongoDB query to find the top 5 restaurants with the highest average score for each cuisine type, along with their average scores.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $group: {

\_id: { cuisine: "$cuisine", name: "$name" },

avgScore: { $avg: "$grades.score" }

}},

{ $sort: { "\_id.cuisine": 1, avgScore: -1 } },

{ $group: {

\_id: "$\_id.cuisine",

topRestaurants: { $push: { name: "$\_id.name", avgScore: "$avgScore" } }

}},

{ $project: {

topRestaurants: { $slice: ["$topRestaurants", 5] }

}}

]);

86. Write a MongoDB query to find the top 5 restaurants in each borough with the highest number of "A" grades.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $match: { "grades.grade": "A" } },

{ $group: {

\_id: { borough: "$borough", name: "$name" },

aCount: { $sum: 1 }

}},

{ $sort: { "\_id.borough": 1, aCount: -1 } },

{ $group: {

\_id: "$\_id.borough",

topRestaurants: { $push: { name: "$\_id.name", aCount: "$aCount" } }

}},

{ $project: {

topRestaurants: { $slice: ["$topRestaurants", 5] }

}}

]);

87. Write a MongoDB query to find the borough with the highest number of restaurants that have a grade of "A" and a score greater than or equal to 90.   
**Ans** : db.restaurants.aggregate([

{ $unwind: "$grades" },

{ $match: {

"grades.grade": "A",

"grades.score": { $gte: 90 }

}},

{ $group: { \_id: "$borough", count: { $sum: 1 } } },

{ $sort: { count: -1 } },

{ $limit: 1 }

]);

**Practical 8:** Database Connectivity:

Do CRUD with MongoDB and ExpressJS

**Practical 9:** Session and Cookies Demo :

Session management

Make use of cookies for storing client data

**Practical 10:** Login Exercise :

Do Authentication exercise with ExpressJS

**Practical 10:** Setting up DevOPS for Project :

Use JIRA, Github, Jenkins and Docker to set the working environment for Project work

**Practical 12:** APIs :

Host Student API and check it.