Dev Ghildiyal **Roll No. 18**

Practical 1

Aim: Write a program to implement MongoDB data models.

Q1. Create a database named as empdb with collection name emp.

1) **Embedded data model:**

Code:

```
use empdb
db.createCollection("emp")
db.emp.insert({
  name: "Dev",
  age: 19,
  email: "dev@gmail.com",
  detail: [{contact:1234567890, familyMember:
"five"},{contact:0987456123,familyMember:"Three"}]
})
```

Output:

```
1 use empdb
2 db.createCollection("emp")
3 db.emp.insert({
4   name: "Dev",
5   age: 19,
6   email: "dev@gmail.com",
7   detail: [{contact:1234567890, familyMember: "five"},{contact:0987456123,familyMember:"Three"}]
8 })
9
```

```
Output:

switched to db empdb
{ "ok" : 1 }
WriteResult({ "nInserted" : 1 })
```

2) **Normalized data model:**

Code:

```
db.emp.insert({
  name: "Dev",
  age: 19,
  website: "empCompany.com"
})
db.emp.find()
```

Output:

```
db.emp.insert({
  name: "Dev",
  age: 20,
  website: "empCompany.com"
})
db.emp.find()
```

```
WriteResult({ "nInserted" : 1 })
{ "_id" :
ObjectId("65e4a20911615bc0f4c9c5c5"),
"name" : "dev", "age" : 19, "website" :
"empCompany.com" }
```

Dev Ghildiyal **Roll No. 18**

Practical 2

Aim: Write a program to implement CRUD operations on MongoDB.

1) Create operation:

Code:

```
use student1
db.createCollection("student1")
```

Output:

```
use student1
db.createCollection("student1")
```

```
Output:

switched to db student1
{ "ok" : 1 }
```

i. insert()

Code:

```
db.student1.insert({
  name: "aamina",
  age: "19",
  roll_no: "221"
})
```

Output:

```
db.student1.insert({
  name: "aamina",
  age: "19",
  roll_no: "221"
})
```

```
Output:

WriteResult({ "nInserted" : 1 })
```

ii. insertMany()

Code:

```
db.student1.insertMany([{
  name: "aamina",
  age: "19",
  roll_no: "221"},
  {
    name: "priyanka",
    age: "19",
    roll_no: "222"},
  {
    name: "karishma",
    age: "20",
    roll_no: "223"}
```

])

Output:

```
db.student1.insertMany([{
  name: "aamina",
  age: "19",
  roll_no: "221"},
  {
    name: "priyanka",
    age: "19",
    roll_no: "222"},
  {
    name: "karishma",
    age: "20",
    roll_no: "223"}
])
```

Output:

```
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6579ae8f5cb560abc70c8e46"),
                ObjectId("6579ae8f5cb560abc70c8e47"),
                ObjectId("6579ae8f5cb560abc70c8e48")
        ]
}
```

2) Read operation:

i. find()

Code:

db.students.find()

Output:

```
db.students.find()
```

```
students> [
  {
    _id: ObjectId("657ac638dee309ef03c6b9ae"),
    name: 'aamina',
    age: '19',
    roll_no: '221'
  },
  {
    _id: ObjectId("657ac638dee309ef03c6b9af"),
    name: 'priyanka',
    age: '19',
    roll_no: '222'
  },
  {
    _id: ObjectId("657ac638dee309ef03c6b9b0"),
    name: 'karishma',
    age: '20',
    roll_no: '223'
  }
]
students>
```

ii. findOne()

Code:

db.students.findOne({name: "aamina"})

Output:

```
1 db.students.findOne({name: "aamina"})
```

```
students> {
  _id: ObjectId("657ac6f49c949ded3b6cfe84"),
  name: 'aamina',
  age: '19',
  roll_no: '221'
}
students>
```

3) Update operation

i. updateOne()

Code:

db.students.updateOne({name: "priyanka"},{$set: {age: "21"}})

Output:

```
1 db.students.updateOne({name: "priyanka"},{$set: {age: "21"}})
```

```
students> {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
students>
```

4) Delete operation

i. deleteOne()

Code:

db.students.deleteOne({name: "aamina"})

db.students.find()

Output:

```
1 db.students.deleteOne({name: "aamina"})
2 db.students.find()
```

```
students> [
  {
    _id: ObjectId("657ac84dc0f8db8306573a8d"),
    name: 'priyanka',
    age: '21',
    roll_no: '222'
  },
  {
    _id: ObjectId("657ac84dc0f8db8306573a8e"),
    name: 'karishma',
    age: '20',
    roll_no: '223'
  }
]
students>
```

Dev Ghildiyal **Roll No.18**

Practical 3

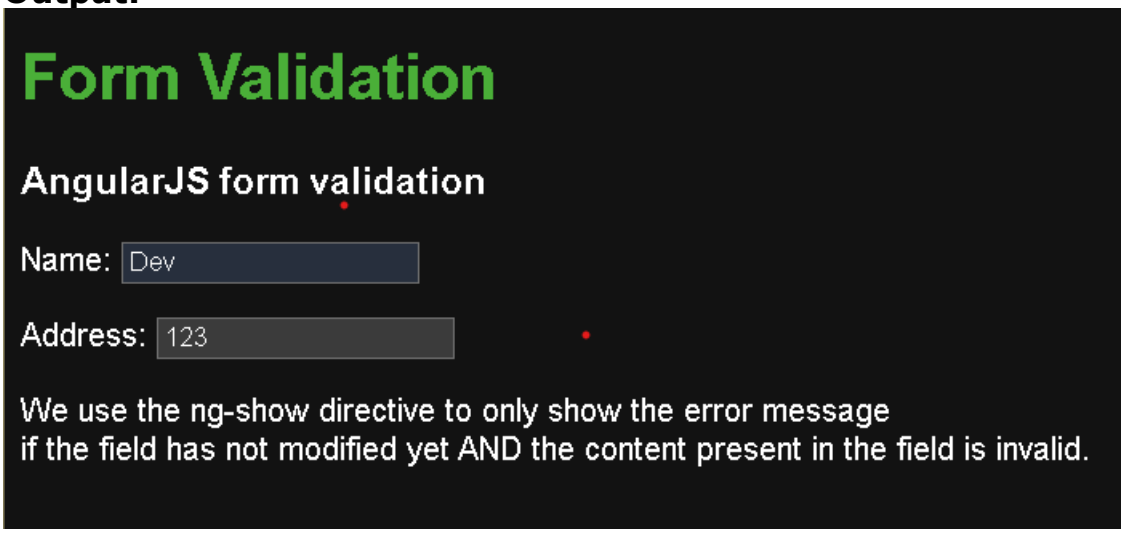
1)Aim: Write a form validation program using AngularJS.

Code:

```
<!doctype html>
<html>
   <head>
      <title>Angular Js Form
      validation</title> <script

src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js
"></script>
      <style>
         body{
            font-family:Arial,Helvetica,sans-serif;
         }
         h1{
            color:green;
         }
      </style>
   </head>

   <body ng-app="">
      <h1>Form Validation</h1>
      <h3>AngularJS form validation</h3>
      <form name="form1">
         <p>Name:
         <input name="username" ng-model="username"
         required> <span ng-show="form1.username.$pristine &&
form1.username.$invalid">
            The name is required.
         </span></p>
         <p>Address:
         <input name="useraddress" ng-model="useraddress"required>

         </p>
      </form>
      <p>
         We use the ng-show directive to only show the error
message<br>if the field has not modified yet AND the content present
in the field is invalid.
```

```
        </p>
    </body>
</html>
```

Output:



Dev Ghildiyal **Roll No.18**

Practical 07

Aim1. Print hello world with void main

```
void main(){
   print("main is the entry point!");
   print("hello world");
 }
```

output:

```
main is the entry point!
hello world
```

Aim2. write a dart program to make the use of comment

```
 int main(){
   var lst=[1,2,3];
   //single line comment
   /*
    * it prints
    * the whole list
    * at once
     */
   print(lst);
   return 0;
 }
```

output:

```
[1, 2, 3]
```

Aim 3. write a dart program to documentation comment (triple line comment)

```
int main(){
  var lst=[1,2,3];
      /// it print the whole list
   /// at once
  print(lst);
  return 0;
}
```

Aim 4. write a dart program to create a variable and display it

```
void main() {
  var a=12;
 const pi=3.14;
 print(a);
  print(pi);
}
```

output:

```
12
3.14
```

Aim 5. Datatypes in dart

```
void main() {
  int num1=2;
 double num2=1.5;
 print (num1);
 print(num2);
```

```
  var a1=num.parse("1");
  var b1=num.parse("2.34");
  var c1=a1+b1;
  print("Product = ${c1}");
}
```

output:

```
2
1.5
Product = 3.34
```

Aim 6. write a dart program to create a list and display it

```
void main() {
  var list=[1,2,3,4];
  print(list);
}
```

output:

```
[1, 2, 3, 4]
```

Aim 7. WAP to create mapping and display

```
void main() {
  var mapping={"id":37,'name':'shraddha'};
  print(mapping);
}
```

output:

```
{id: 37, name: shraddha}
```

Aim 8. write a dart program to print even no. with for loop(range between 1-10)

```
void main() {
  for (int i = 1;i<=10;i++){
```

```
    if(i%2==0){
      print(i);
    }
  }
  }
```

Output:

```
2
4
6
8
10
```

Aim 9:WAP to add two variables using function

```
void main() {
  add(3,4);
}
void add(int a,int b){
  int c;
  c=a+b;
  print(c);
  }
```

Output:

-

Aim 10. write a dart program to demonstrate assignment operator

```
void main() {
  // Assignment operators
  double a = 5;
  double b = 3;
```

```
  // = (Assignment)
  double result = a + b;
  print("Result after '=' operator: $result");

  // += (Add and assign)
  result += 2;
  print("Result after '+=' operator: $result");

  // -= (Subtract and assign)
  result -= 1;
  print("Result after '-=' operator: $result");

  // *= (Multiply and assign)
  result *= 3;
  print("Result after '*=' operator: $result");

  // /= (Divide and assign)
  result /= 2;
  print("Result after '/=' operator: $result");

  // %= (Remainder and assign)
  result %= 4;
  print("Result after '%=' operator: $result");
}
```

```
Result after '=' operator: 8
Result after '+=' operator: 10
Result after '-=' operator: 9
Result after '*=' operator: 27
Result after '/=' operator: 13.5
Result after '%=' operator: 1.5
```

Aim 11. Write a dart program to implement getter and setter.

code:

```
class Employee{
  String name="";
  String get emp_name{
    return name;
  }
  void set emp_name(String name){
    this.name=name;
  }
  void result(){
    print(name);
  }
}
void main(){
  Employee emp=new Employee();
  emp.name="priyanka";
  emp.result();
}
```

Output:

```
priyanka
```

Dev Ghildliyal **Roll No.18**

Practical 08

Aim 1. write a dart program to implement single inheritance

```
class Bird{
  void fly()
  {
    print("The bird can fly");
  }
}
class Parrot extends Bird{
  void speak(){
    print("The parrot can speak");
  }
}
void main(){
  Parrot p=new Parrot();
  p.speak();
  p.fly();
}
```

OUTPUT:

```
The parrot can speak
The bird can fly
```

Aim 2. write a dart program to implement multilevel inheritance

```
class Bird{
  void fly()
  {
```

```
    print("The bird can fly");
  }
}
class Parrot extends Bird{
  void speak(){
    print("The parrot can speak");
  }
}
class Eagle extends Parrot{
  void vision(){
    print("The eagle has a very sharp vision");
  }
}
void main(){
  Eagle e=new Eagle();
  e.speak();
  e.fly();
  e.vision();
}
```

OUTPUT:

```
The parrot can speak
The bird can fly
The eagle has a very sharp vision
```

Aim 3. write a dart program to implement hierarchical inheritance

```
  class Person{
    void disName(String name)
    {
```

```
    print(name);
  }
  void disAge(int age){
    print (age);
  }
}
class Peter extends Person{
  void disBranch(String nationality){
    print(nationality);
  }
}
class James extends Person{
  void result(String result){
    print(result);
  }
}
void main(){
  James j=new James();
  j.disName("James");
  j.disAge(24);
  j.result("Passed");
  Peter p=new Peter();
  p.disName("Peter");
  p.disAge(21);
  p.disBranch("Computer Science");
}
```

OUTPUT:

```
James
24
Passed
Peter
21
Computer Science
```

Aim 4. write a dart program to implement abstract method

```
abstract class Person{
    void disInfo();
  }
  class Boy extends Person{
    void disInfo(){
      print("my name is peter ");
    }
  }
  class Girl extends Person{
    void disInfo(){
      print("My name is Shraddha");
    }
  }
  void main(){
    Boy b=new Boy();
    Girl g=new Girl();
    b.disInfo();
    g.disInfo();
  }
```

```
my name is peter
My name is Shraddha
```