PERIYAR UNIVERSITY

(NAAC 'A++' Grade with CGPA 3.61 (Cycle - 3)
State University - NIRF Rank 56 - State Public University Rank 25
SALEM - 636 011, Tamil Nadu, India.

CENTRE FOR DISTANCE AND ONLINE EDUCATION (CDOE)

MASTER OF COMPUTER APPLICATIONS
SEMESTER - I

ELECTIVE – I: DATA ENGINEERING AND MANAGEMENT LAB

(Candidates admitted from 2024 onwards)

PERIYAR UNIVERSITY

CENTRE FOR DISTANCE AND ONLINE EDUCATION (CDOE)
MCA 2024 admission onwards

Elective Course – I LAB

DATA ENGINEERING AND MANAGEMENT LAB

Prepared by:

Centre for Distance and Online Education (CDOE)
Periyar University
Salem – 636011.

SYLLABUS

DATA ENGINEERING AND MANAGEMENT LAB

COURSE OBJECTIVES

- To acquire basic scripting knowledge in MongoDB
- To learn CRUD Operation on MongoDB database
- To comprehend MongoDB using DbVisualizer
- To be familiar with Zoho CRM features
- To customize your application using Zoho CRM

LIST OF EXPERIMENTS

1. Write a script to create a MongoDB database and perform insert operation
2. Write a MongoDB script to perform query operations
3. Write a MongoDB Script to perform update operations
4. Write a MongoDB Script to update documents with aggregation pipeline
5. Write a MongoDB script to delete single and multiple documents
6. Write a MongoDB script to perform string aggregation operations
7. Design a Data Model for MongoDB using DbVisualizer
8. Perform CRUD operations using DbVisualizer
9. Create a Zoho CRM account and organize your Tasks, Meetings and Deals
10. Create and maintain a project using Zoho CRM features

COURSE OUTCOMES

On the successful completion of the course, students will be able to:

CO1	Comprehend the scripting knowledge in MongoDB and perform basic operations in shell prompt	**K1- K6**
CO2	Implement, Create, Read, Update and Delete Operations on MongoDB databases	
CO3	Analyze MongoDB using DbVisualizer	
CO4	Assess Zoho CRM features for managing the customer relationships	
CO5	Create a customized application in Zoho CRM	

K1- Remember, K2- Understand, K3- Apply, K4- Analyze, K5- Evaluate, K6- Create

MAPPING WITH PROGRAMME OUTCOMES

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10
CO1	H	H	H	H	H	M	H	H	H	M
CO2	H	H	H	H	H	H	H	H	H	H
CO3	H	M	H	H	M	H	M	H	H	H
CO4	H	H	H	M	H	H	H	L	H	H
CO5	H	H	H	H	M	H	H	H	H	H

H- High; M-Medium; L-Low

CONTENTS

1 PERFORM INSERT OPERATION

AIM:

TO WRITE A PROGRAM FOR PERFORM INSERT OPERATION

ALGORITHM:

- **Step 1:** Start the process
- **Step 2: C**reate a database
- **Step 3: S**elect the database using this command (use database name)
- **Step 4:** Insert the data only one by using the command insertone()
- **Step 5:** Insert the many data at a time by using the command insertmany()
- **Step 6:** Stop the process

SOURCE CODE:

```
P01> db.stuinfo.insertOne(
{"_id" : 1, Name:'Dhanush', Age:'21', Course:'MCA'}
)
```

```
P01> db.stuinfo.insertMany([
{"_id" : 2,Name : 'Divya', Age : '22', Course:'MSC CS'},
{"_id" : 3,Name : 'Shalini', Age : '20', Course: 'MSC DS'},
{"_id" : 4,Name : 'Hariharan', Age : '24', Course: 'MSc IT'}
])
```

OUTPUT:

➢ Insert Single Documents:

```
{ acknowledged: true, insertedIds: { '0': 1 } }
```

➢ Insert Many Documents:

```
{ acknowledged: true, insertedIds: { '0': 2, '1': 3, '2': 4 } }
```

➢ Database:

```
[
  { _id: 1, Name: 'Dhanush', Age: '21', Course: 'MCA' },
  { _id: 2, Name: 'Divya', Age: '22', Course: 'MSC CS' },
  { _id: 3, Name: 'Shalini', Age: '20', Course: 'MSC DS' },
  { _id: 4, Name: 'Hariharan', Age: '24', Course: 'MSc IT' }
]
```

RESULT:

THUS THE ABOVE QUERY HAS BEEN EXECUTED SUCESSFULLY

2 PERFORM QUERY OPERATION

AIM:

TO WRITE A PROGRAM FOR PERFORM QUERY OPERATION

ALGORITHM:

- **Step 1:** Start the process
- **Step 2: C**reate a database
- **Step 3: S**elect the database using this command (use database name)
- **Step 4:** Insert the many data at a time by using the command insertmany()
- **Step 5:** Find the data in the database by the find() command under many term
- **Step 6:** Stop the process

SOURCE CODE:

```
P02> db.stuinfo.insertMany([
{ "_id": 1, Name: 'Divya', Age: '22', Course: 'MSC CS' },
{ "_id": 2, Name: 'Shalini', Age: '20', Course: 'MSC DS' },
{ "_id": 3, Name: 'Hariharan', Age: '24', Course: 'MSc IT' }
])
```

```
P02> db.stuinfo.find()
P02> db.stuinfo.find({Name:'Shalini'})
P02> db.stuinfo.find({$or:[{Name:'David'},{Age:'20'}]}) P02>
db.stuinfo.find({Name:{$in:["Hariharan","David"]}}) P02> db.stuinfo.find({ Age: { $gt:'21'}})
P02> db.stuinfo.find({ Age: { $lt:'21'}})
P02> db.stuinfo.find({ Name: { $eq:'Shalini'}}) P02> db.stuinfo.find({ Name: {
$nin:['Shalini']}})
```

OUTPUT:

➢ Finding Documents:

```
[
  { _id: 1, Name: 'Divya', Age: '22', Course: 'MSC CS' },
  { _id: 2, Name: 'Shalini', Age: '20', Course: 'MSC DS' },
  { _id: 3, Name: 'Hariharan', Age: '24', Course: 'MSc IT' }
]
```

➢ AND Operator:

:

```
[ { _id: 2, Name: 'Shalini', Age: '20', Course: 'MSC DS' } ]
```

➢ OR Operator:

```
[
  { _id: 1, Name: 'Divya', Age: '22', Course: 'MSC CS' },
  { _id: 2, Name: 'Shalini', Age: '20', Course: 'MSC DS' }
]
```

➢ IN Operator:

```
[
  { _id: 1, Name: 'Divya', Age: '22', Course: 'MSC CS' },
  { _id: 3, Name: 'Hariharan', Age: '24', Course: 'MSc IT' }
]
```

➢ Greater than Operator:

```
[
  { _id: 1, Name: 'Divya', Age: '22', Course: 'MSC CS' },
  { _id: 3, Name: 'Hariharan', Age: '24', Course: 'MSc IT' }
]
```

➢ Less than Operator:

```
[ { _id: 2, Name: 'Shalini', Age: '20', Course: 'MSC DS' } ]
```

➢ Equal to Operator:

```
[ { _id: 2, Name: 'Shalini', Age: '20', Course: 'MSC DS' } ]
```

➢ NIN Operator:

```
[
  { _id: 1, Name: 'Divya', Age: '22', Course: 'MSC CS' },
  { _id: 3, Name: 'Hariharan', Age: '24', Course: 'MSc IT' }
]
```

RESULT:

THUS THE ABOVE QUERY HAS BEEN EXECUTED SUCESSFULLY

3 PERFORM UPADATE OPERATION

AIM:

TO WRITE A PROGRAM FOR PERFORM UPDATE OPERATION

ALGORITHM:

- **Step 1:** Start the process
- **Step 2: C**reate a database
- **Step 3: S**elect the database using this command (use database name)
- **Step 4:** Insert the many data at a time by using the command insertmany()
- **Step 5:** Update the data in the database by the UPDATE() command under many term
- Step 6: Replace the data by using the command replace()
- **Step** 7**:** Stop the process

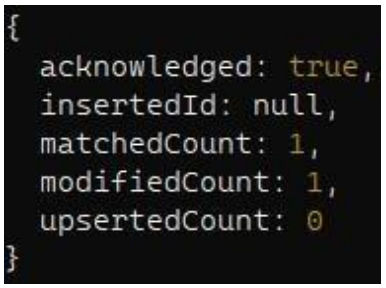
SOURCE CODE:

```
P03> db.stuinfo.insertMany([
{ "_id": 1, Name: 'Divya', Age: '22', Gender:'female',Course:
'MSC CS' },
{ "_id": 2, Name: 'Shalini', Age: '20',Gender:'female', Course:
'MSC DS' },
{ "_id": 3, Name: 'Hariharan', Age: '24',Gender:'Male'}
])
```

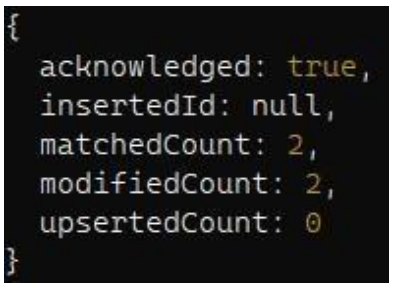
```
P03> db.stuinfo.updateOne({ "_id":3 }, { $set: { Course: 'MSc IT' } })
P03>db.stuinfo.updateMany({Gender:'female'},{$set:{Gender:
'F'}})
P03> db.stuinfo.replaceOne({ Name: 'Dhanusg' }, {  "_id":4, Name: 'Dhanush', Age:
'19',Gender:'Male', Course: 'MCA' }, { upsert: true })
```

OUTPUT:

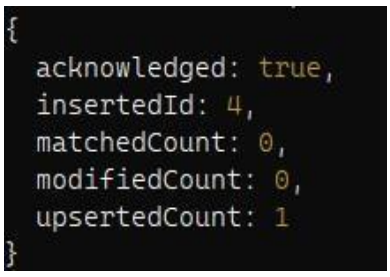
➢ Update Single Document:



➢ Update Multiple Document:



➢ Replace One Document:



RESULT:

THUS THE ABOVE QUERY HAS BEEN EXECUTED SUCESSFULLY

4 UPDATE DOCUMENTS WITH AGGREGATION PIPELINE

AIM:

TO WRITE A PROGRAM FOR PERFORM UPDATE DOCUMENTS WITH AGGREGATION PIPELINE OPERATION

ALGORITHM:

- **Step 1:** Start the process
- **Step 2: C**reate a database
- **Step 3: S**elect the database using this command (use database name)
- **Step 4:** Insert the many data at a time by using the command insertmany()
- **Step 5:** Aggregate the data in the database by the aggregate() command under many term
- **Step 6:** Stop the process

SOURCE CODE:

```
P04> db.stuinfo.insertMany([
{ "_id": 1, Name: 'Divya', Age: '22', Gender:'female',Course:
'MSC CS' },
{ "_id": 2, Name: 'Shalini', Age: '20',Gender:'female', Course:
'MSC DS' },
{ "_id": 3, Name: 'Hariharan', Age:
'24',Gender:'Male',Course:'MSC IT'}
])
```

```
P04> db.stuinfo.aggregate([{ $addFields: { College: 'Periyar
University' } }])
P04> db.stuinfo.aggregate([{ $set: { Location :'Salem'}}]) P04> db.stuinfo.aggregate([{$project : {
Name : 1 , Course : 1
}}])
P04> db.stuinfo.aggregate([{$unset: ['Age']}])
P04> db.stuinfo.aggregate([{$replaceRoot: { newRoot:
{Name_and_Course:{$concat: ['$Name',' & ','$Course']}}}}])
```

OUTPUT:

➢ Add Fields:


```
[
  {
    _id: 1,
    Name: 'Divya',
    Age: '22',
    Gender: 'female',
    Course: 'MSC CS',
    College: 'Periyar University'
  },
  {
    _id: 2,
    Name: 'Shalini',
    Age: '20',
    Gender: 'female',
    Course: 'MSC DS',
    College: 'Periyar University'
  },
  {
    _id: 3,
    Name: 'Hariharan',
    Age: '24',
    Gender: 'Male',
    Course: 'MSC IT',
    College: 'Periyar University'
  }
]
```


➢ Set field:


```
[
  {
    _id: 1,
    Name: 'Divya',
    Age: '22',
    Gender: 'female',
    Course: 'MSC CS',
    Location: 'Salem'
  },
  {
    _id: 2,
    Name: 'Shalini',
    Age: '20',
    Gender: 'female',
    Course: 'MSC DS',
    Location: 'Salem'
  },
  {
    _id: 3,
    Name: 'Hariharan',
    Age: '24',
    Gender: 'Male',
    Course: 'MSC IT',
    Location: 'Salem'
  }
]
```

➢ Project:

```
[
  { _id: 1, Name: 'Divya', Course: 'MSC CS' },
  { _id: 2, Name: 'Shalini', Course: 'MSC DS' },
  { _id: 3, Name: 'Hariharan', Course: 'MSC IT' }
]
```

➢ Unset:

```
{ _id: 1, Name: 'Divya', Gender: 'female', Course: 'MSC CS' },
{ _id: 2, Name: 'Shalini', Gender: 'female', Course: 'MSC DS' },
{ _id: 3, Name: 'Hariharan', Gender: 'Male', Course: 'MSC IT' }
```

➢ Replace Root:

```
[
  { Name_and_Course: 'Divya & MSC CS' },
  { Name_and_Course: 'Shalini & MSC DS' },
  { Name_and_Course: 'Hariharan & MSC IT' }
]
```

RESULT:

THUS THE ABOVE QUERY HAS BEEN EXECUTED SUCESSFULLY

5 DELETE SINGLE AND MULTIPLE DOUCUMENTS

AIM:

TO WRITE A PROGRAM FOR PERFORM DELETE OPERATION

ALGORITHM:

- **Step 1:** Start the process
- **Step 2: C**reate a database
- **Step 3: S**elect the database using this command (use database name)
- **Step 4:** Insert the many data at a time by using the command insertmany()
- **Step 5:** Delete the data in the database by the delete() command under two terms deleteOne() and deleteMany()
- **Step 6:** Stop the process

SOURCE CODE:

```
P05> db.stuinfo.insertMany([
{ "_id": 1, Name: 'Divya', Age: '22', Gender:'female',Course:
'MSC CS' ,Location:'Salem'},
{ "_id": 2, Name: 'Shalini', Age: '20',Gender:'female', Course:
'MSC DS', Location:'Kovai' },
 { "_id": 3, Name: 'Hariharan', Age: '24', Gender:'Male',Course:'MSC IT',
Location:'Salem'},
{"_id":4,Name:'Dhanush',Age:'19',Gender:'Male',Course:'MCA', Location:'Erode'}
])
```

```
P05> db.stuinfo.deleteOne({"_id": 2})
P05> db.stuinfo.deleteMany({Location : 'Salem'}) P05> db.stuinfo.remove({ })
```

OUTPUT:

- Delete Single Documents:

```
{ acknowledged: true, deletedCount: 1 }
```

- Delete Many Documents:

```
{ acknowledged: true, deletedCount: 2 }
```

- Remove entire Documents:

```
{ acknowledged: true, deletedCount: 1 }
```

RESULT:

THUS THE ABOVE QUERY HAS BEEN EXECUTED SUCESSFULLY

6 PERFORM STRING AGGREGATION OPERATIONS

AIM:

TO WRITE A PROGRAM FOR PERFORM STRING AGGREGATION OPERATION

ALGORITHM:

- **Step 1:** Start the process
- **Step 2: C**reate a database
- **Step 3: S**elect the database using this command (use database name)
- **Step 4:** Insert the many data at a time by using the command insertmany()
- **Step 5:** Aggregate the data in the database by the aggregate() command under the terms concatenate, upper case, lower case, split, and find length functions
- **Step 6:** Stop the process

SOURCE CODE:

```
P06> db.stuinfo.insertMany([
{ "_id": 1, Name: 'Divya', Age: '22', Gender:'female',Course:
'MSC CS' ,Location:'Salem'},
{ "_id": 2, Name: 'Shalini', Age: '20',Gender:'female', Course:
'MSC DS', Location:'Kovai' },
 { "_id": 3, Name: 'Hariharan', Age: '24', Gender:'Male',Course:'MSC IT',
Location:'Salem'},
{"_id":4,Name:'Dhanush',Age:'19',Gender:'Male',Course:'MCA', Location:'Erode'}
])
```

```
P06> db.stuinfo.aggregate([{$project:{Concatenate_Value:
{$concat: ['$Name','and','$Age']}}}])
P06> db.stuinfo.aggregate([{$project:{Split_Value:{$split: ['$Name','&']}}}])
P06> db.stuinfo.aggregate([{$project:{UpperCase_Value:
{$toUpper:['$Name']}}}])
P06> db.stuinfo.aggregate([{$project:{LowerCase_Value:
{$toLower:['$Name']}}}])
P06> db.stuinfo.aggregate([{$project:{FindLength_Value:
{$strLenCP: ['$Name']}}}])
```

OUTPUT:

➢ Concatenate Value:

```
[
  { _id: 1, Concatenate_Value: 'Divyaand22' },
  { _id: 2, Concatenate_Value: 'Shaliniand20' },
  { _id: 3, Concatenate_Value: 'Hariharanand24' },
  { _id: 4, Concatenate_Value: 'Dhanushand19' }
]
```

➢ Split Value:

```
[
  { _id: 1, Split_Value: [ 'Divya' ] },
  { _id: 2, Split_Value: [ 'Shalini' ] },
  { _id: 3, Split_Value: [ 'Hariharan' ]
  { _id: 4, Split_Value: [ 'Dhanush' ] }
]
```

➢ Upper Case Value:

```
[
  { _id: 1, UpperCase_Value: 'DIVYA' },
  { _id: 2, UpperCase_Value: 'SHALINI' },
  { _id: 3, UpperCase_Value: 'HARIHARAN' },
  { _id: 4, UpperCase_Value: 'DHANUSH' }
]
```

➢ Lower Case Value:

```
[
  { _id: 1, LowerCase_Value: 'divya' },
  { _id: 2, LowerCase_Value: 'shalini' },
  { _id: 3, LowerCase_Value: 'hariharan' },
  { _id: 4, LowerCase_Value: 'dhanush' }
]
```

➢ Find the length:

```
[
  { _id: 1, FindLength_Value: 5 },
  { _id: 2, FindLength_Value: 7 },
  { _id: 3, FindLength_Value: 9 },
  { _id: 4, FindLength_Value: 7 }
]
```

RESULT:

THUS THE ABOVE QUERY HAS BEEN EXECUTED SUCESSFULLY

7 DESIGN DATA MODEL FOR MONGODB USING DB VISUALIZER

AIM:

TO WRITE A PROGRAM FOR DESIGN DATA MODEL FOR MONGODB USING DB VISUALIZER

ALGORITHM:

- **Step 1:** Start the process
- **Step 2:** Install MongoDB and DB Visualizer
- **Step 3:** Connect to MongoDB using DB Visualizer
  - Open DB Visualizer.
  - Create a new connection and select MongoDB as the database type.
  - Enter the connection details (host, port, and authentication details if needed).
- **Step 4:** Create the Data Model
  - Determine the collections (tables) and their relationships.
  - Define the fields (columns) and their data types for each collection.

  Let's design a simple data model for an e-commerce application with the following collections:

    - **Users**
    - **Products**
    - **Orders**

- **Step 5:** Stop the process

SOURCE CODE:

```
from pymongo import MongoClient
from uuid import uuid4
from datetime import datetime

# Connect to MongoDB
client = MongoClient('localhost', 27017)
db = client['ecommerce']

# Insert Users
users = db['users']
users.insert_one({
    "user_id": str(uuid4()),
    "name": "John Doe",
    "email": "john@example.com",
    "password": "password123",
    "address": {
        "street": "123 Main St",
        "city": "Anytown",
        "state": "CA",
        "zip": "12345"
    }
})

# Insert Products
products = db['products']
products.insert_many([
    {
        "product_id": str(uuid4()),
        "name": "Laptop",
        "description": "A powerful laptop",
        "price": 999.99,
        "category": "Electronics",
        "stock_quantity": 50
    },
    {
        "product_id": str(uuid4()),
        "name": "Headphones",
        "description": "Noise-cancelling headphones",
        "price": 199.99,
        "category": "Electronics",
        "stock_quantity": 200
```

```
    }
])

# Insert Orders
orders = db['orders']
orders.insert_one({
    "order_id": str(uuid4()),
    "user_id": users.find_one({"name": "John Doe"})['user_id'],
    "order_date": datetime.now(),
    "total_amount": 1199.98,
    "products": [
        {
            "product_id": products.find_one({"name": "Laptop"})['product_id'],
            "quantity": 1,
            "price": 999.99
        },
        {
            "product_id": products.find_one({"name": "Headphones"})['product_id'],
            "quantity": 1,
            "price": 199.99
        }
    ],
    "status": "Pending"
})

print("Data inserted successfully")
```

OUTPUT:

➢ Users Collection

```json
{
  "user_id": "550e8400-e29b-41d4-a716-446655440000",
  "name": "John Doe",
  "email": "john@example.com",
  "password": "password123",
  "address": {
    "street": "123 Main St",
    "city": "Anytown",
    "state": "CA",
    "zip": "12345"
  }
}
```

➢ Products Collection

```json
{
  "product_id": "550e8400-e29b-41d4-a716-446655440001",
  "name": "Laptop",
  "description": "A powerful laptop",
  "price": 999.99,
  "category": "Electronics",
  "stock_quantity": 50
},
{
  "product_id": "550e8400-e29b-41d4-a716-446655440002",
  "name": "Headphones",
  "description": "Noise-cancelling headphones",
  "price": 199.99,
  "category": "Electronics",
  "stock_quantity": 200
}
```

➢ Orders Collection

```json
{
  "order_id": "550e8400-e29b-41d4-a716-446655440003",
  "user_id": "550e8400-e29b-41d4-a716-446655440000",
  "order_date": "2024-06-25T10:00:00Z",
  "total_amount": 1199.98,
  "products": [
    {
      "product_id": "550e8400-e29b-41d4-a716-446655440001",
      "quantity": 1,
      "price": 999.99
    },
    {
      "product_id": "550e8400-e29b-41d4-a716-446655440002",
      "quantity": 1,
      "price": 199.99
    }
  ],
  "status": "Pending"
}
```

RESULT:

THUS THE ABOVE QUERY HAS BEEN EXECUTED SUCESSFULLY

8 CRUD OPERATIONS USING DBVISUALIZER

AIM:

TO WRITE A PROGRAM FOR CRUD OPERATIONS USING DBVISUALIZER

ALGORITHM:

- **Step 1:** Start the process
- **Step 2: Create (Insert) Operation:** Open DB Visualizer and connect to your MongoDB database.
- **Step 3:** Navigate to the database where you want to insert the data.
- **Step 4:** Right-click on the collection where you want to insert the data and select “Open SQL Commander”.
- **Step 5:** Write the insert command in JavaScript (MongoDB shell syntax) and execute it.
- **Step 6: Read (Query) Operation:** Navigate to the collection you want to query.
- **Step 7:** Right-click on the collection and select “Open SQL Commander”
- **Step 8:** Write the query command in JavaScript and execute it.
- **Step 9: Update Operation:** Navigate to the collection you want to update.
- **Step 10:** Right-click on the collection and select “Open SQL Commander”
- **Step 11:** Write the update command in JavaScript and execute it.
- **Step 12: Delete Operation:** Navigate to the collection you want to delete from.
- **Step 13:** Right-click on the collection and select “Open SQL Commander”
- **Step 14:** Write the delete command in JavaScript and execute it.
- **Step 15:** Stop the process

SOURCE CODE:

Create (Insert) Operation:

```
db.users.insertOne({
  "user_id": "550e8400-e29b-41d4-a716-446655440000",
  "name": "John Doe",
  "email": "john@example.com",
  "password": "password123",
  "address": {
    "street": "123 Main St",
    "city": "Anytown",
    "state": "CA",
    "zip": "12345"
  }
})
```

Read (Query) Operation:

```
db.users.find({
  "name": "John Doe"
})
```

Update Operation:

```
db.users.updateOne(
  { "name": "John Doe" },
  {
    $set: { "email": "john.doe@example.com" }
  }
)
```

Delete Operation:

```
db.users.deleteOne({
  "name": "John Doe"
})
```

OUTPUT:

Insert User

```javascript
db.users.insertOne({
  "user_id": "550e8400-e29b-41d4-a716-446655440000",
  "name": "John Doe",
  "email": "john@example.com",
  "password": "password123",
  "address": {
    "street": "123 Main St",
    "city": "Anytown",
    "state": "CA",
    "zip": "12345"
  }
})
```

Query User

```javascript
db.users.find({
  "name": "John Doe"
})
```

Update User

```javascript
db.users.updateOne(
  { "name": "John Doe" },
  {
    $set: { "email": "john.doe@example.com" }
  }
)
```

Delete User

```javascript
db.users.deleteOne({
  "name": "John Doe"
})
```

RESULT:

THUS THE ABOVE QUERY HAS BEEN EXECUTED SUCESSFULLY

9 ZOHO CRM ACCOUNT AND ORGANZIE YOUR TASKS, MEETINGS AND DEALS

AIM:

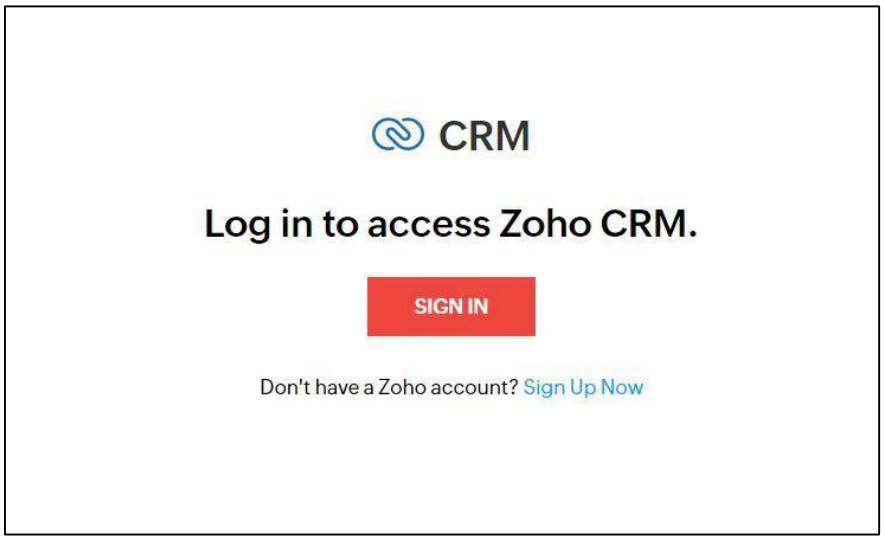
TO WRITE A PROGRAM FOR PERFORM INSERT OPERATION

ALGORITHM:

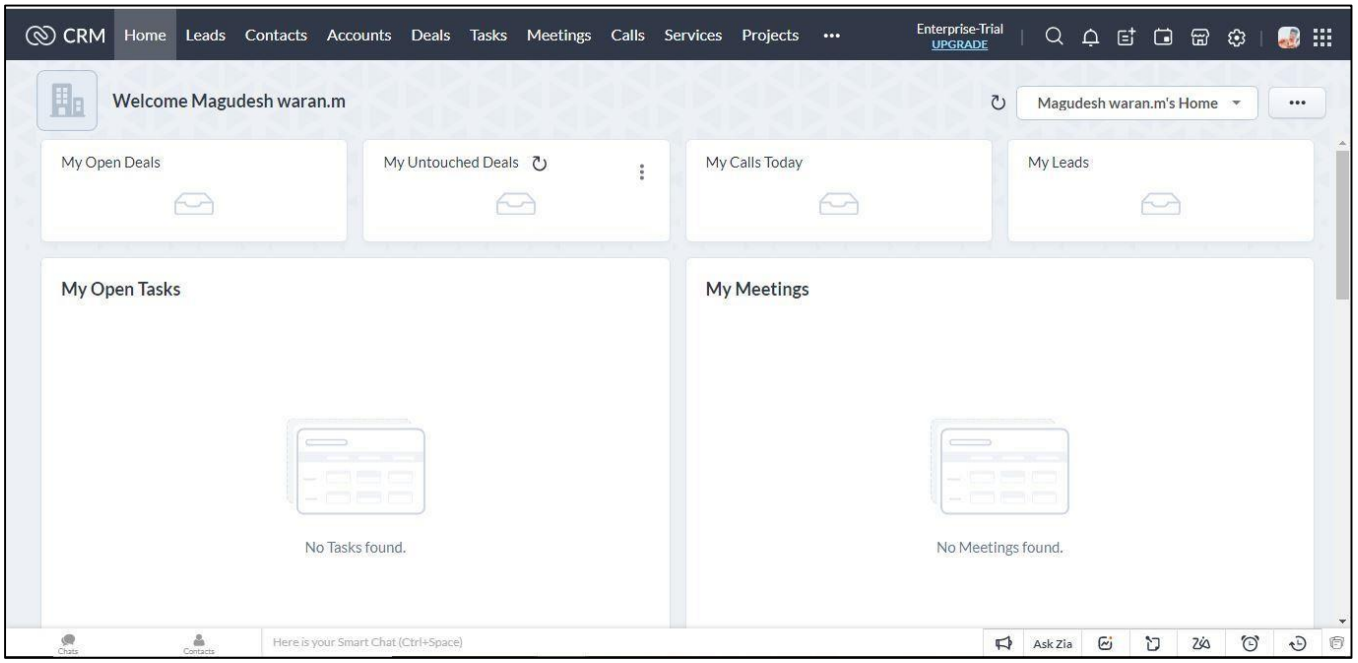
- **Step 1:** Start the process
- **Step 2: C**reate a ZOHO CRM ACCOUNT AT THEIR OFFICIAL WEBSITE
- **Step 3:** Go to the home page after creating the account and select the tasks at the top of the page and customize it according to the user needs
- **Step 4:** Then select the deals and customize it also and save the progress
- **Step 5:** Then select the meetings option to set the details that asks for the meeting and save the process
- **Step 6:** Stop the process

SOURCE CODE:

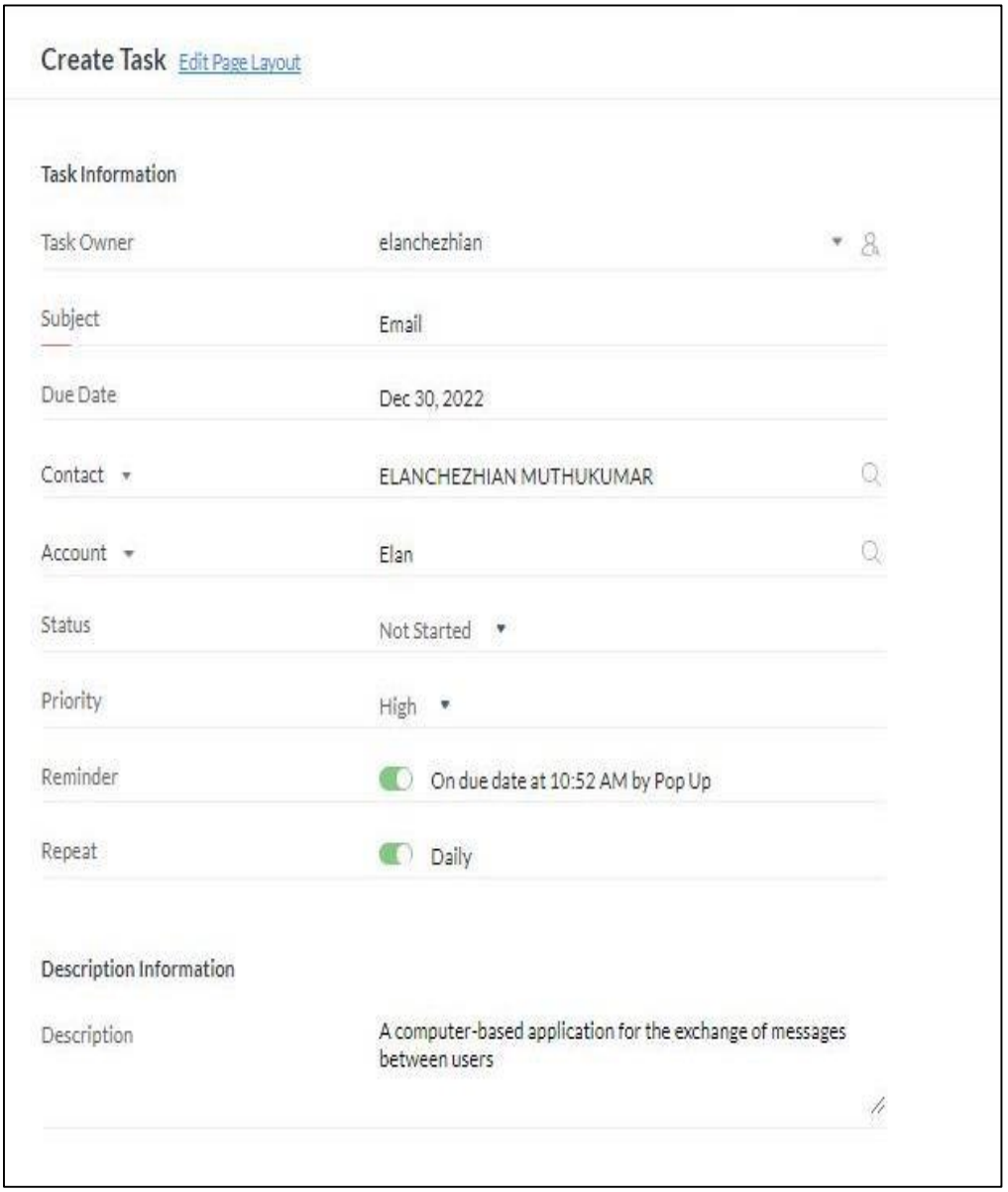
- Creating Zoho CRM account:



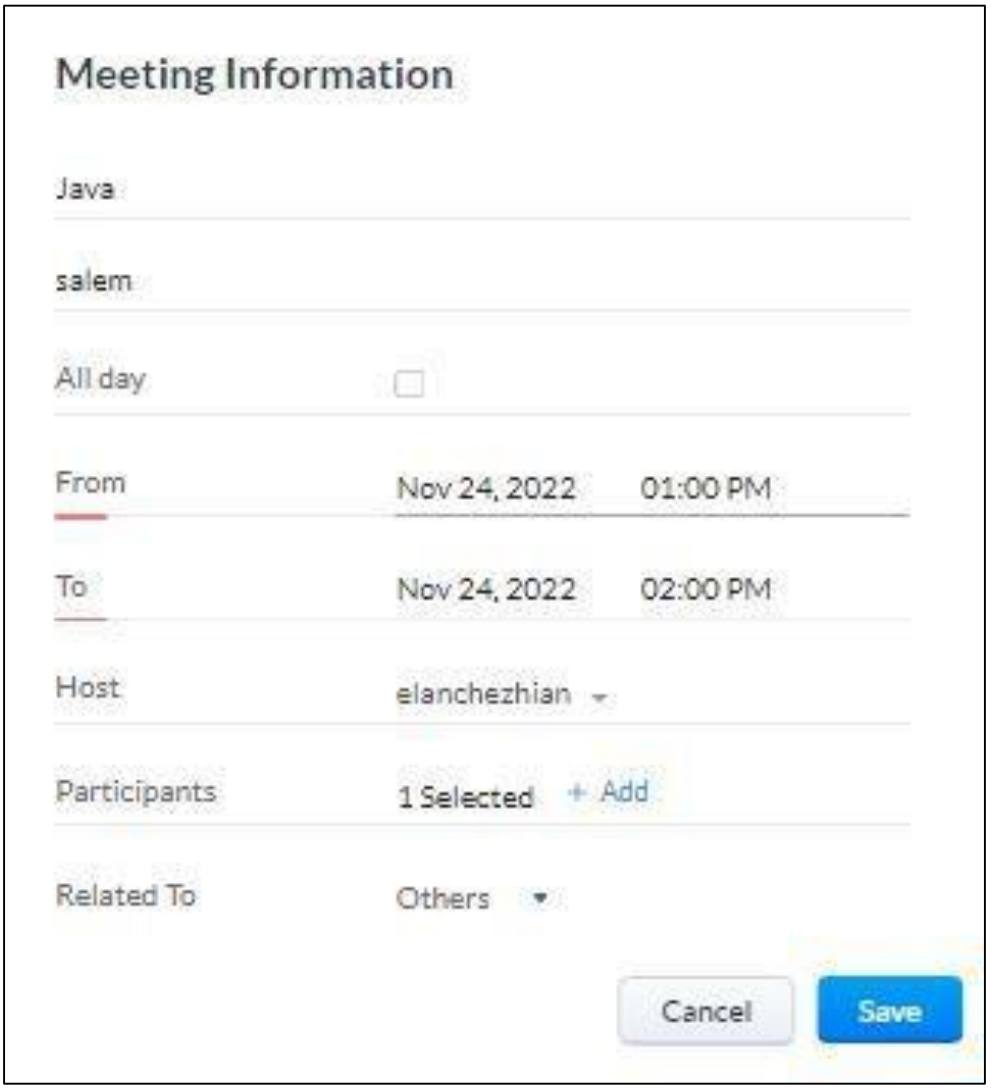
- Home:



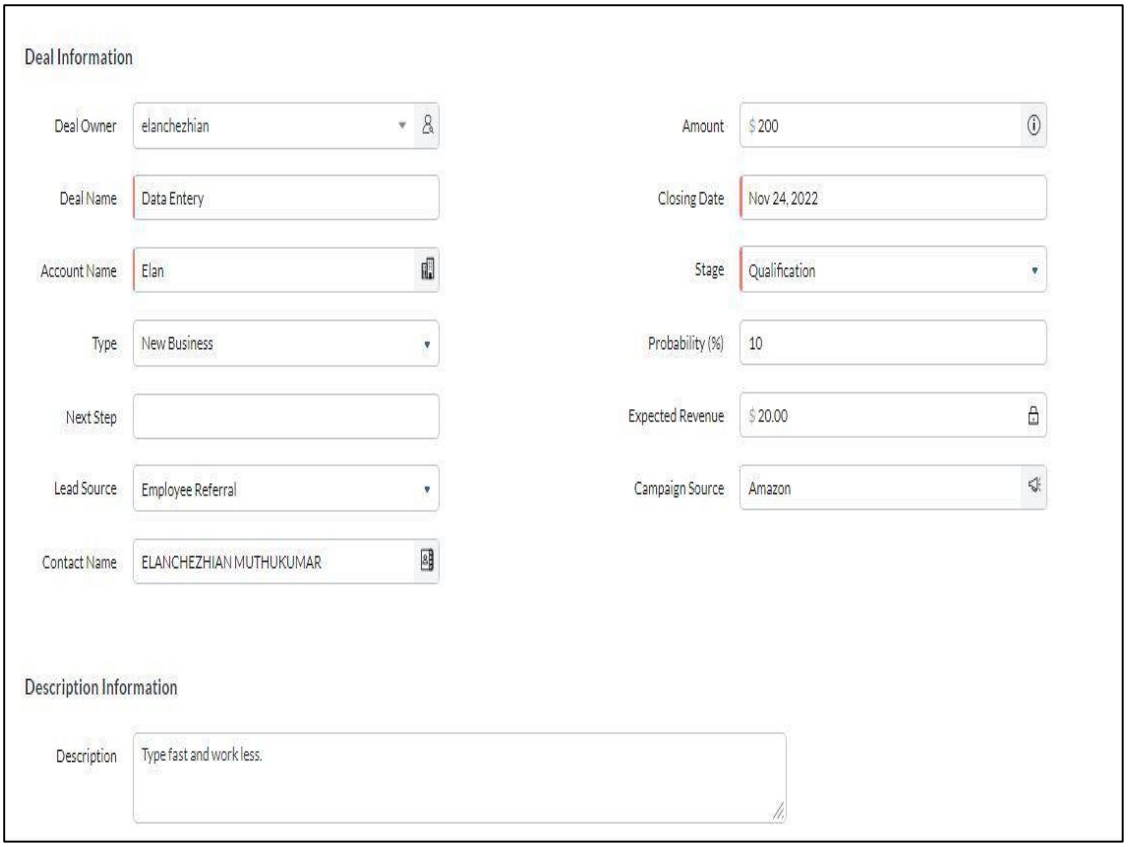
➢ Tasks:


Create Task
Edit Page Layout
Task Information
Task Owner
elanchezhian
Subject
Email
Due Date
Dec 30, 2022
Contact
ELANCHEZHIAN MUTHUKUMAR
Account
Elan
Status
Not Started
Priority
High
Reminder
On due date at 10:52 AM by Pop Up
Repeat
Daily
Description Information
Description
A computer-based application for the exchange of messages between users

➢ **Meetings:**

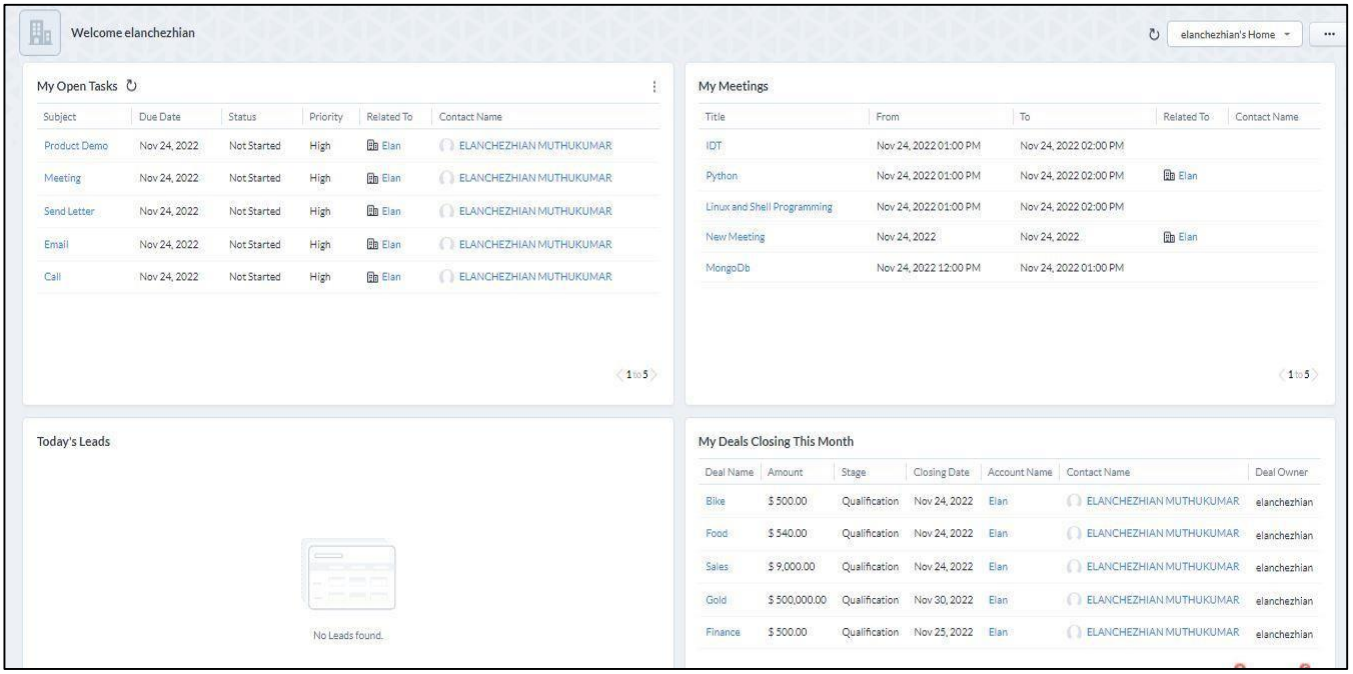

Meeting Information
Java
salem
All day
From
Nov 24, 2022
01:00 PM
To
Nov 24, 2022
02:00 PM
Host
elanchezhian
Participants
1 Selected
+ Add
Related To
Others
Cancel
Save

➢ Deals:


Deal Information
Deal Owner
elanchezhian
Amount
$ 200
Deal Name
Data Entery
Closing Date
Nov 24, 2022
Account Name
Elan
Stage
Qualification
Type
New Business
Probability (%)
10
Next Step
Expected Revenue
$ 20.00
Lead Source
Employee Referral
Campaign Source
Amazon
Contact Name
ELANCHEZHIAN MUTHUKUMAR
Description Information
Description
Type fast and work less.

OUTPUT:

➢ Task , Meetings and Deals:



RESULT:

THUS THE ABOVE QUERY HAS BEEN EXECUTED SUCESSFULLY

10 PROJECT USING ZOHO CRM FEATURES

AIM:

TO WRITE A PROGRAM FOR PERFORM INSERT OPERATION

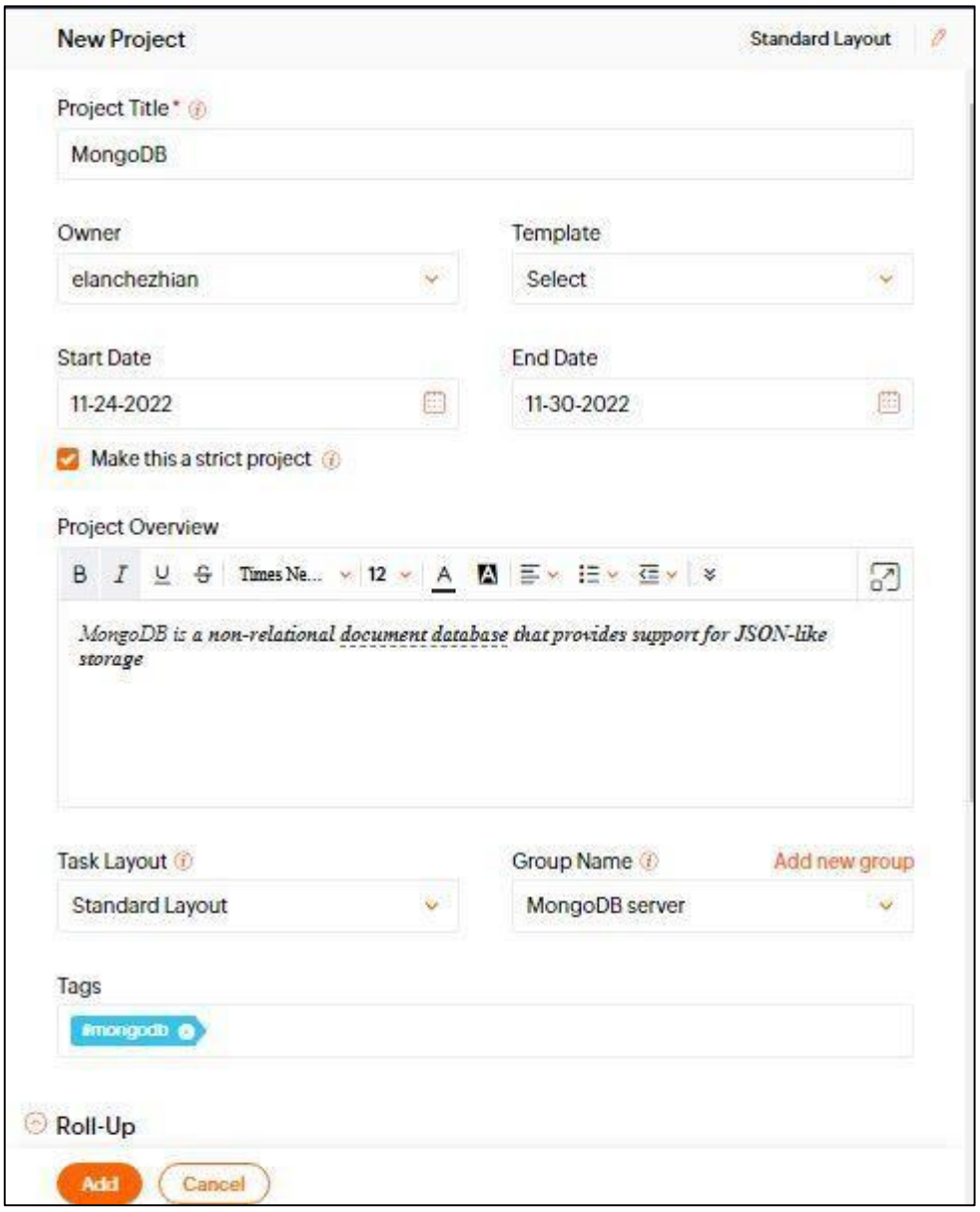
ALGORITHM:

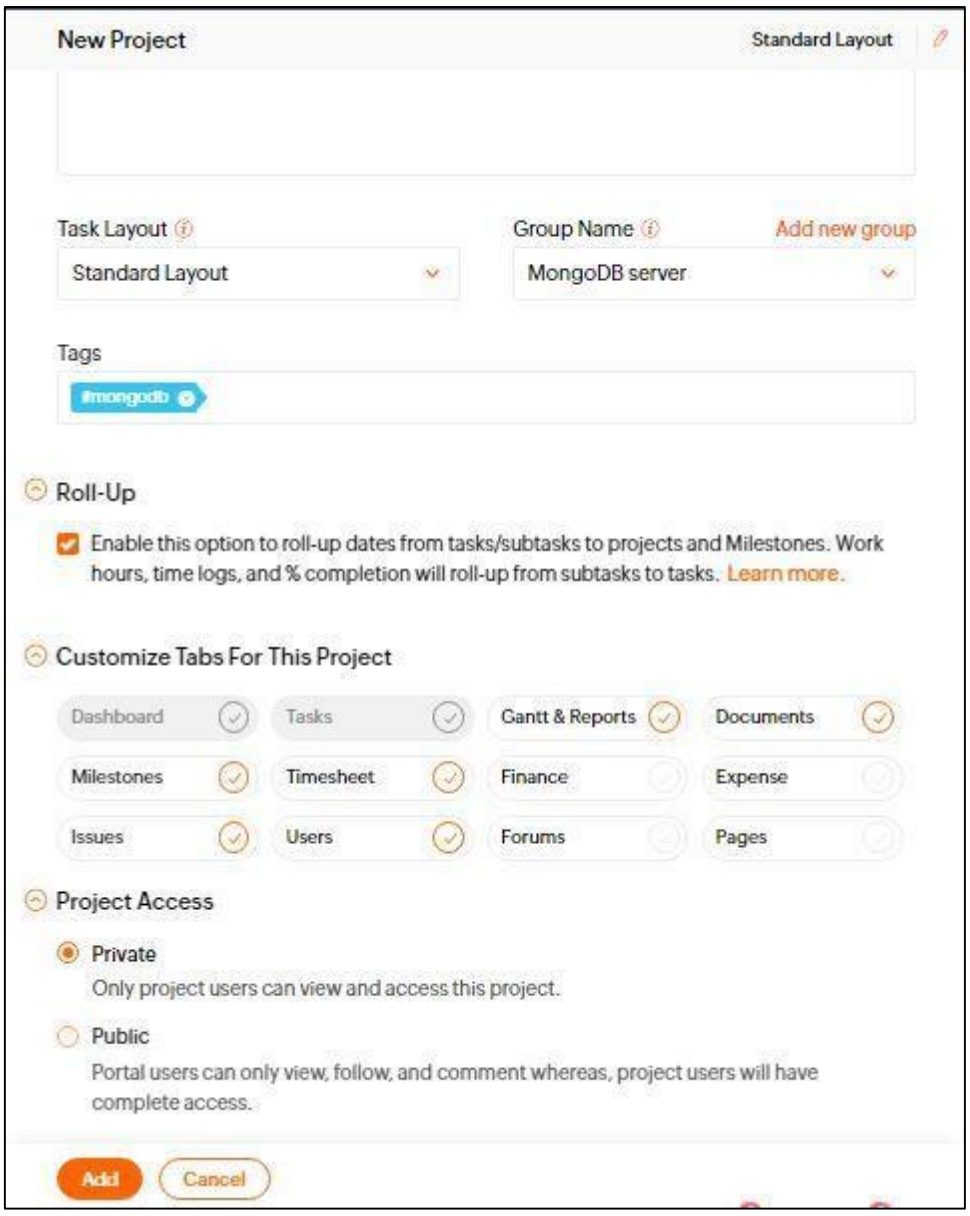
- **Step 1:** Start the process
- **Step 2: C**reate a ZOHO CRM ACCOUNT AT THEIR OFFICIAL WEBSITE
- **Step 3:** Go to the home page after creating the account and click the new project
- **Step 4:** Then enter the details and customize it and save the progress
- **Step 6:** Stop the process

SOURCE CODE AND OUTPUT:

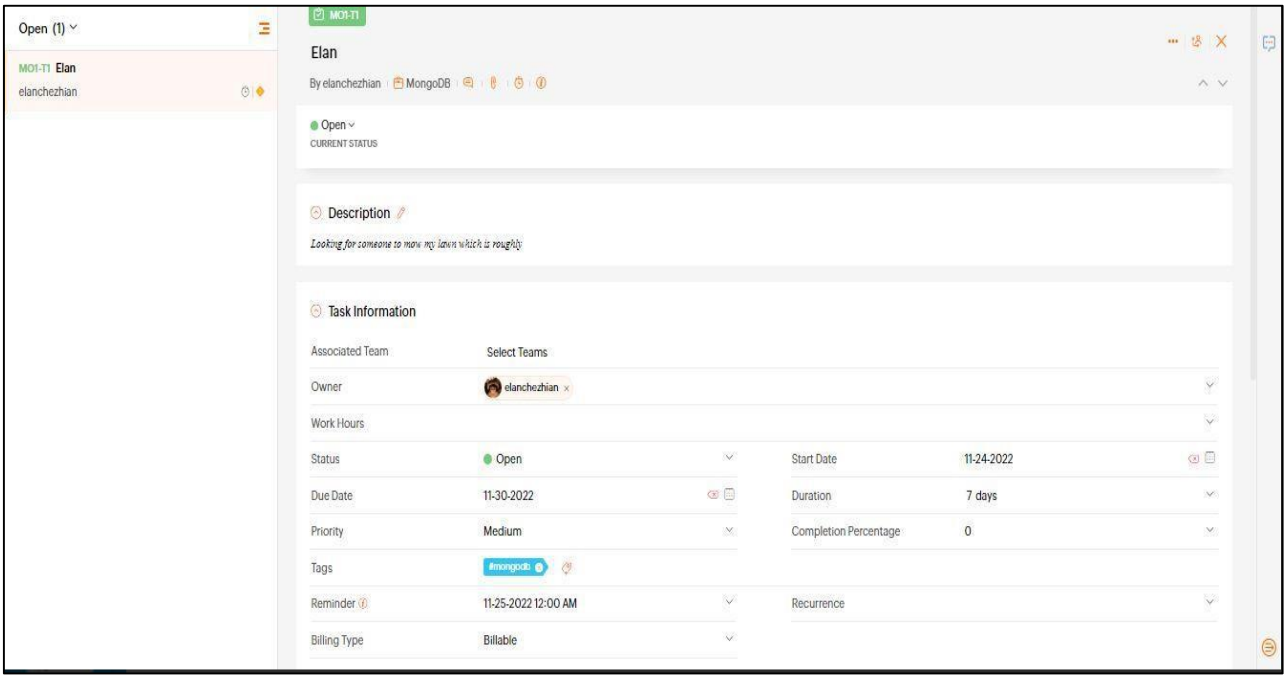
➢ To create a project:

❖ Click **New Project.**




New Project
Standard Layout
Task Layout
Standard Layout
Group Name
Add new group
MongoDB server
Tags
#mongodb
Roll-Up
Enable this option to roll-up dates from tasks/subtasks to projects and Milestones. Work hours, time logs, and % completion will roll-up from subtasks to tasks. Learn more.
Customize Tabs For This Project
Dashboard
Tasks
Gantt & Reports
Documents
Milestones
Timesheet
Finance
Expense
Issues
Users
Forums
Pages
Project Access
Private
Only project users can view and access this project.
Public
Portal users can only view, follow, and comment whereas, project users will have complete access.
Add
Cancel

➢ **Final Project:**



RESULT:

THUS THE ABOVE QUERY HAS BEEN EXECUTED SUCESSFULLY

Reference Books:

1. Martin Kleppmann, “Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems”, 1st Edition, O’Reilly Media, Inc., 2017.
2. Paul Crickard, “Data Engineering with Python: Work with massive datasets to design data models and automate data pipelines using Python”, First Edition, Packt Publishing, 2020.
3. Ralph Kimball and Margy Ross, “The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling”, Third Edition, Wiley & Sons, Inc., 2013.
4. James Densmore, “Data Pipelines Pocket Reference: Moving and Processing Data for Analytics O’Reilly Media, Inc., 2021.
5. Kristin Briney, “Data Management for Researchers: Organize, Maintain and Share Your Data for Research Success”, Pelagic Publishing. 2015.

Website References:

1. https://www.kaggle.com/
2. https://www.coursera.org/courses?query=data%20engineering
3. https://www.dataengineeringweekly.com/
4. https://aws.amazon.com/big-data/blog/
5. https://spark.apache.org/