**Sr.No: 1**

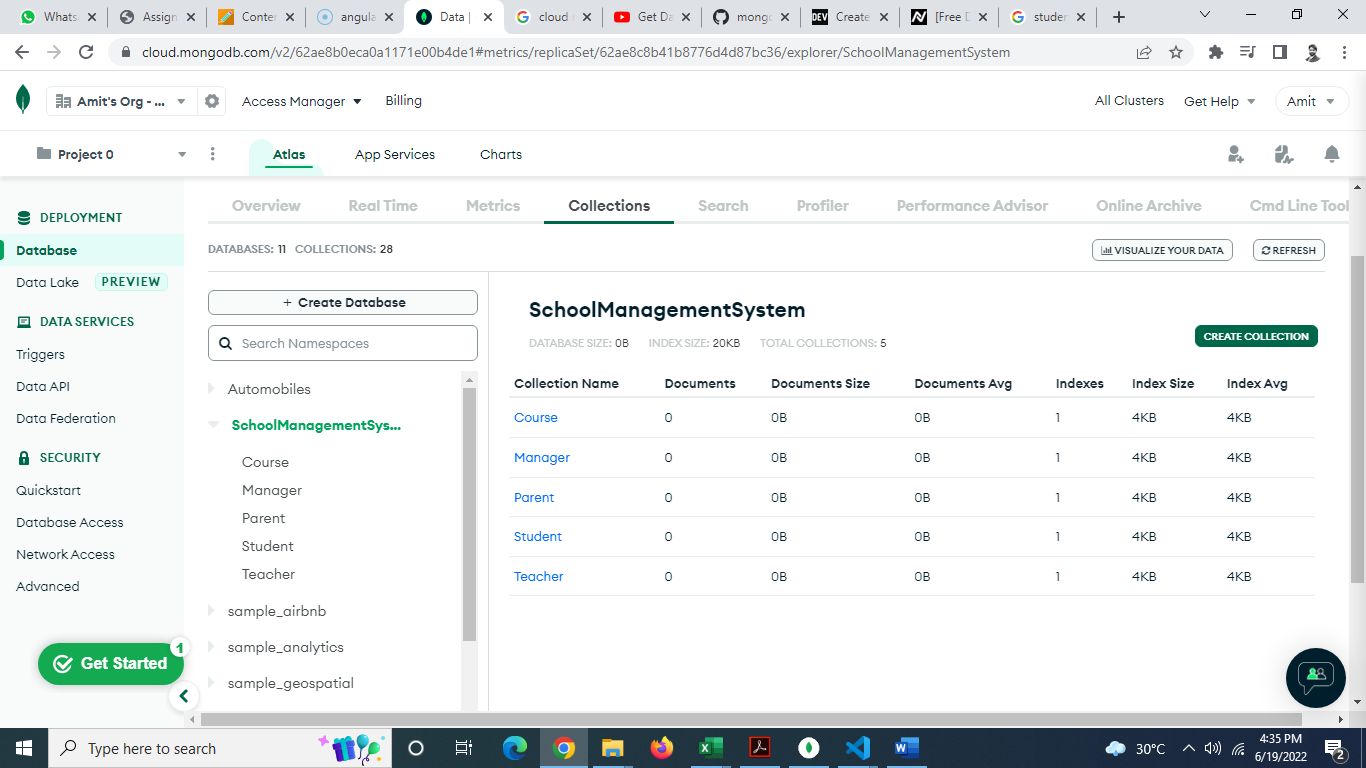
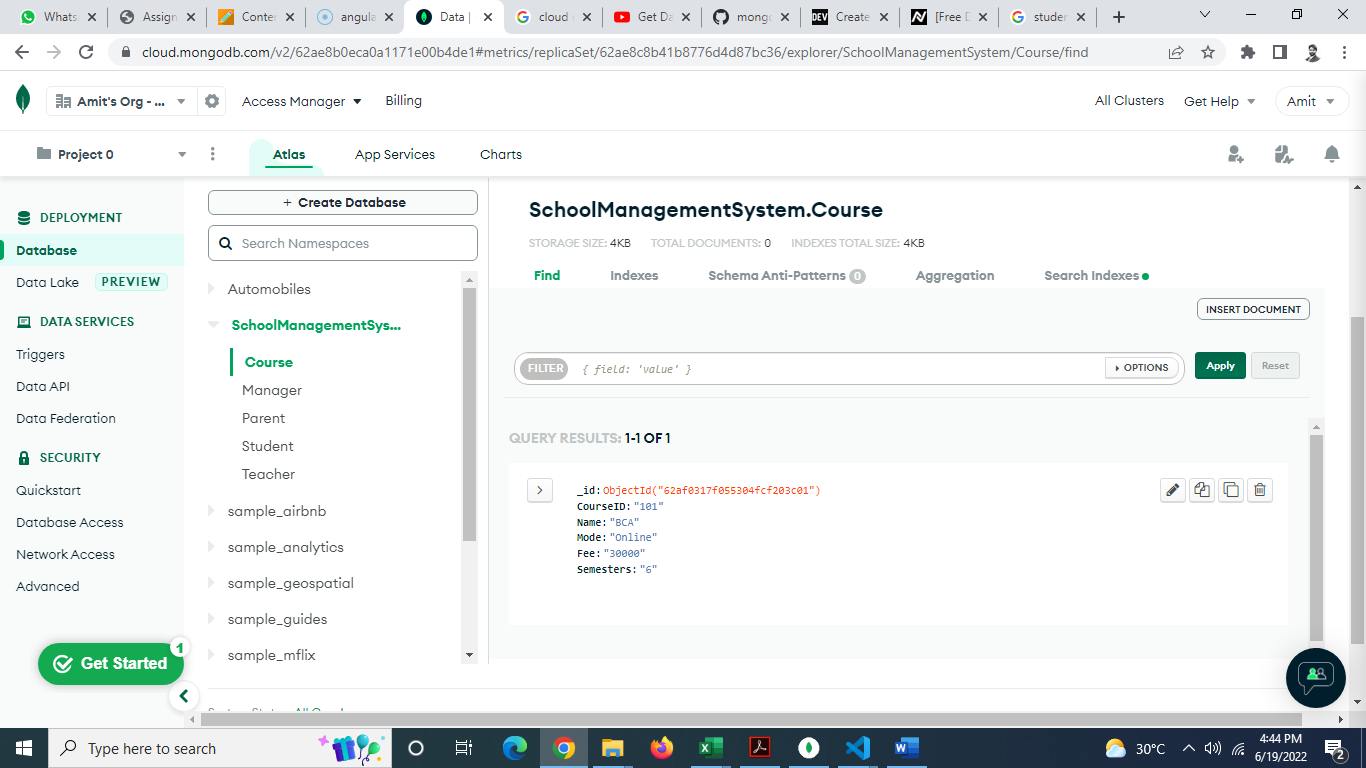
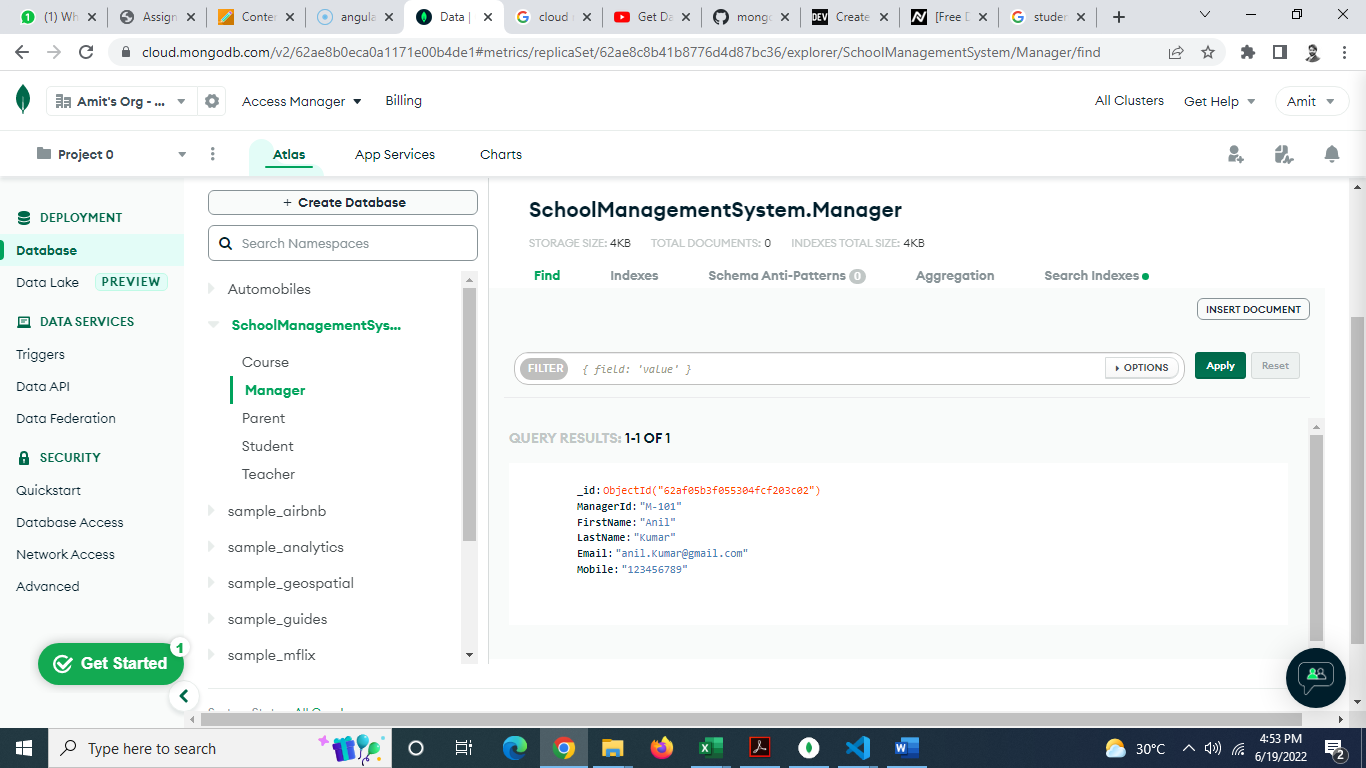
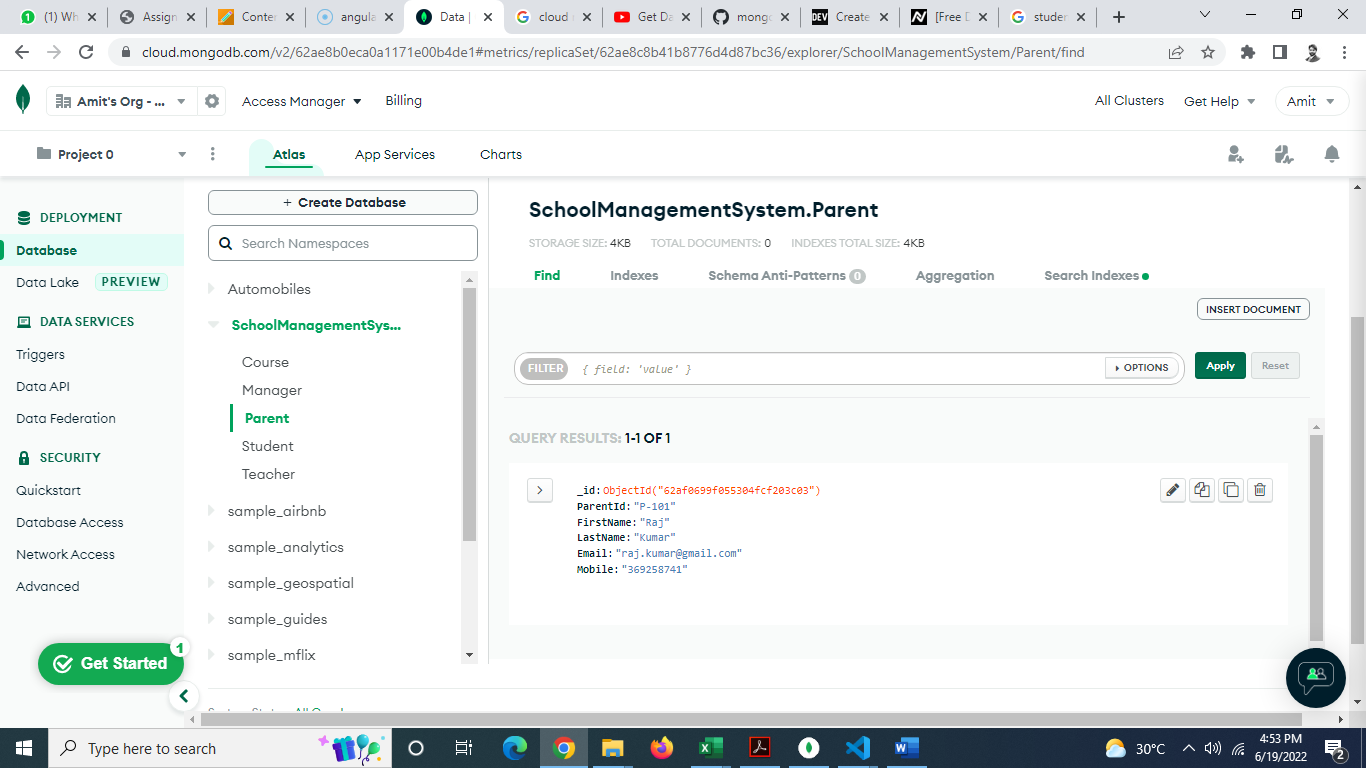
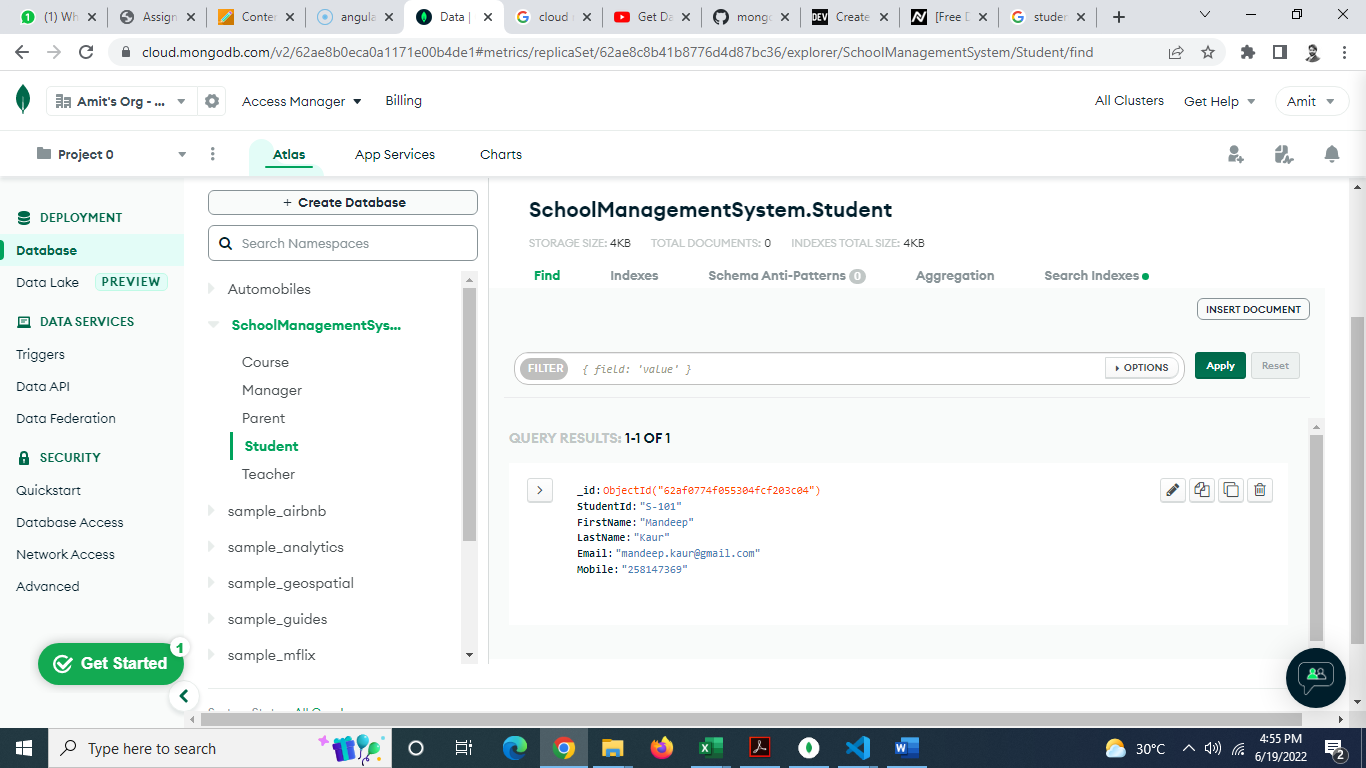
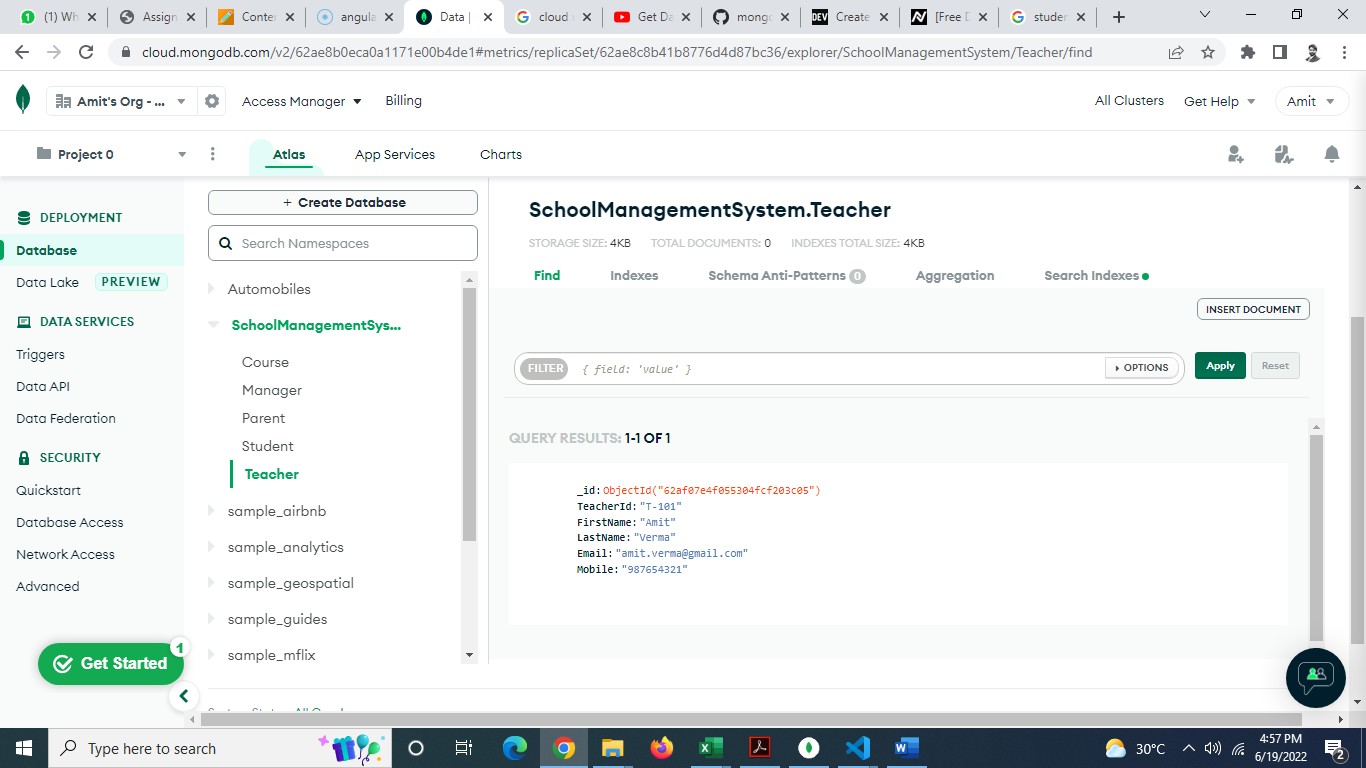
**School Management Application**

It manages most of the tasks which needs many efforts and are time consuming and can be done with very less efforts and also take less time. It also saves environment by not using so much paper.

It has five tables: Students, Teachers, Course, Parents and Managers.

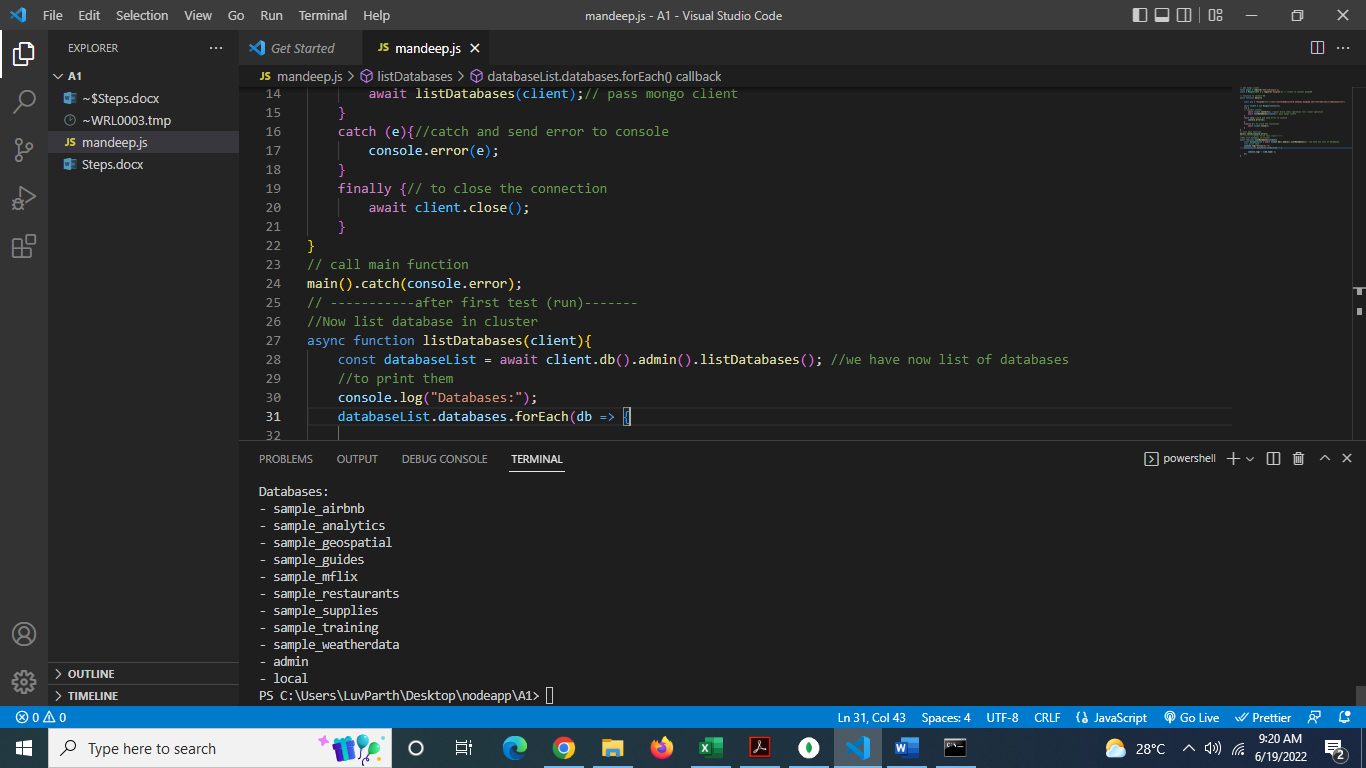
**Sr.No: 2**

1. Database SchoolManagementSystem with tables Course, Manager, Parent, Student and Teacher.

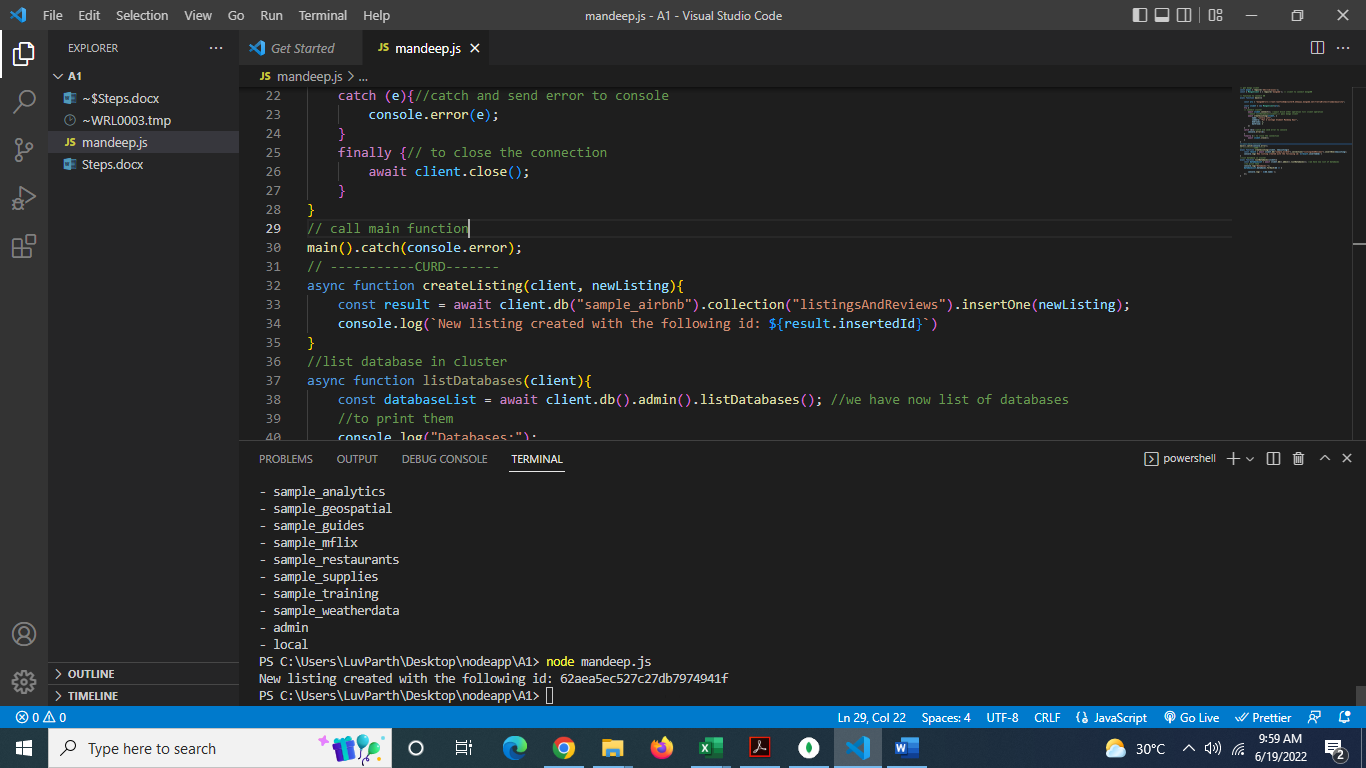
1. 
2. 
3. 
4. 
5. 
6. 

**Sr.No: 3**

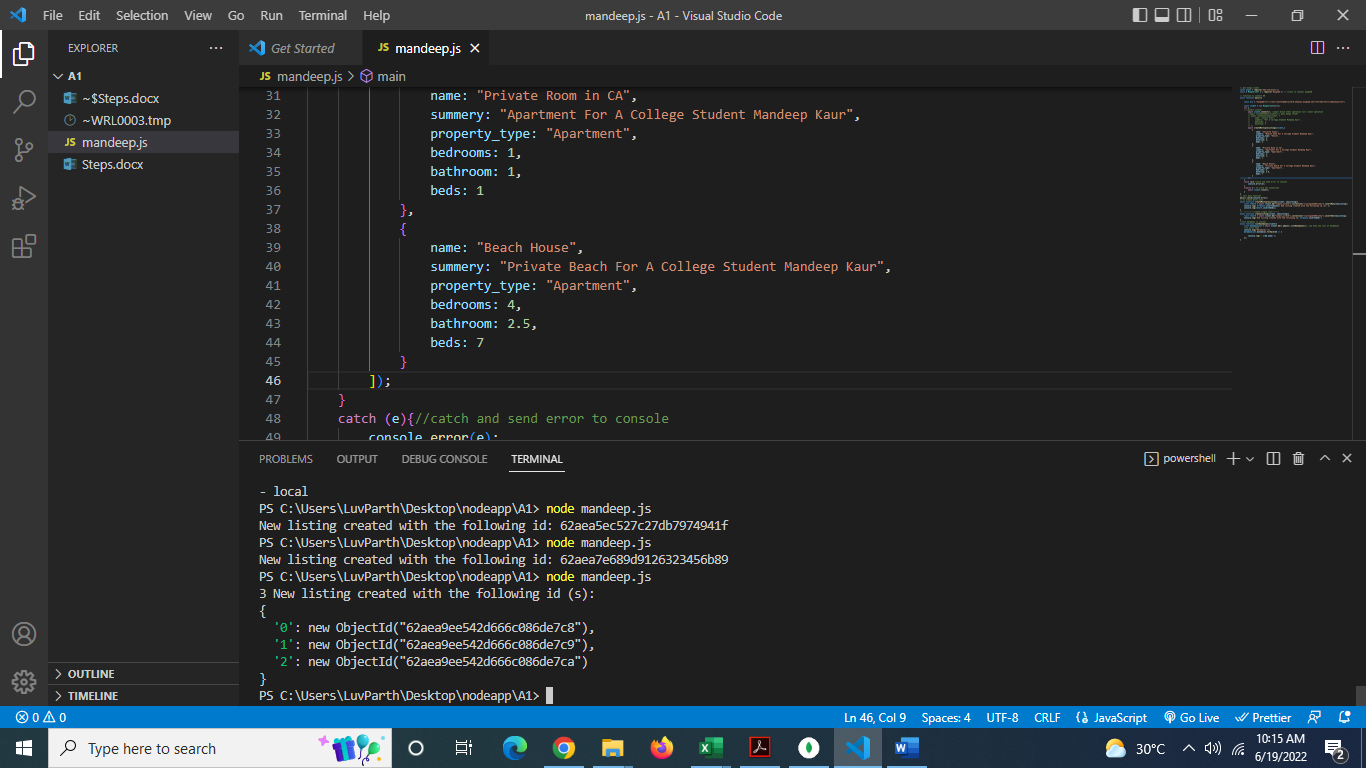
1. Open V-Code
2. Check in the terminal node version if installed or not with node -v command
3. Install mongo DB with the command npm install mongodb hit enter
4. We can check mongodb version with npm list mongodb command
5. Create a Mongo Database account or signin with a google email (gmail id) https://cloud.mongodb.com
6. Choose Free and Create a cluster
7. load Sample database set
8. enter your system ip to access cluster
9. To find ip: First, click on your Start Menu and type cmd in the search box and press enter. A black and white window will open where you will type ipconfig /all and press enter. There is a space between the command ipconfig and the switch of /all. Your ip address will be the IPv4 address.
10. Choose connection method connect your application
11. Select your driver and version node.js
12. Copy connection
13. Database setup now
14. Create script Mandeep.js and run script
15. // get mongo client
16. const async = require('hbs/lib/async');
17. const { MongoClient } = require('mongodb'); // client to connect mongoDB
18. // function to connect DB
19. async function main(){
20. const uri = "mongodb+srv://root:PASSWORD@cluster0.ehkayyu.mongodb.net/?retryWrites=true&w=majority";
21. const client = new MongoClient(uri);
22. try {
23. // connect cluster
24. await client.connect(); //await block other operation till client operation
25. await listDatabases(client);// pass mongo client
26. }
27. catch (e){//catch and send error to console
28. console.error(e);
29. }
30. finally {// to close the connection
31. await client.close();
32. }
33. }
34. // call main function
35. main().catch(console.error);
36. // -----------after first test (run)-------
37. //Now list database in cluster
38. async function listDatabases(client){
39. const databaseList = await client.db().admin().listDatabases(); //we have now list of databases
40. //to print them
41. console.log("Databases:");
42. databaseList.databases.forEach(db => {
44. console.log(`- ${db.name}`);
45. });
46. }
47. Command node Mandeep.js in terminal



1. CURD Create operation
2. For new listing updated code
3. // get mongo client
4. const async = require('hbs/lib/async');
5. const { MongoClient } = require('mongodb'); // client to connect mongoDB
6. // function to connect DB
7. async function main(){
8. const uri = "mongodb+srv://root:PASSWORD@cluster0.ehkayyu.mongodb.net/?retryWrites=true&w=majority";
9. const client = new MongoClient(uri);
10. try {
11. // connect cluster
12. await client.connect(); //await block other operation till client operation
13. //await listDatabases(client);// pass mongo client
14. await createListing(client, {
15. name: "Lovely Loft",
16. summery: "For A College Student Mandeep Kaur",
17. bedrooms: 2,
18. bathroom: 2
19. })
20. }
21. catch (e){//catch and send error to console
22. console.error(e);
23. }
24. finally {// to close the connection
25. await client.close();
26. }
27. }
28. // call main function
29. main().catch(console.error);
30. // -----------CURD-------
31. async function createListing(client, newListing){
32. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").insertOne(newListing);
33. console.log(`New listing created with the following id: ${result.insertedId}`)
34. }
35. //list database in cluster
36. async function listDatabases(client){
37. const databaseList = await client.db().admin().listDatabases(); //we have now list of databases
38. //to print them
39. console.log("Databases:");
40. databaseList.databases.forEach(db => {
42. console.log(`- ${db.name}`);
43. });
44. }



1. Code for Create a multiple data entry in a collection
2. // get mongo client
3. const async = require('hbs/lib/async');
4. const { MongoClient } = require('mongodb'); // client to connect mongoDB
5. // function to connect DB
6. async function main(){
7. const uri = "mongodb+srv://root:PASSWORD@cluster0.ehkayyu.mongodb.net/?retryWrites=true&w=majority";
8. const client = new MongoClient(uri);
9. try {
10. // connect cluster
11. await client.connect(); //await block other operation till client operation
12. //await listDatabases(client);// pass mongo client
13. // await createListing(client, {
14. //     name: "Lovely Loft",
15. //     summery: "For A College Student Mandeep Kaur",
16. //     bedrooms: 2,
17. //     bathroom: 2
18. // })
19. await createMultipleListings(client,[
20. {
21. name: "Infinite Views",
22. summery: "Modren Home For A College Student Mandeep Kaur",
23. property\_type: "House",
24. bedrooms: 2,
25. bathroom: 2,
26. beds: 2
27. },
28. {
29. name: "Private Room in CA",
30. summery: "Apartment For A College Student Mandeep Kaur",
31. property\_type: "Apartment",
32. bedrooms: 1,
33. bathroom: 1,
34. beds: 1
35. },
36. {
37. name: "Beach House",
38. summery: "Private Beach For A College Student Mandeep Kaur",
39. property\_type: "Apartment",
40. bedrooms: 4,
41. bathroom: 2.5,
42. beds: 7
43. }
44. ]);
45. }
46. catch (e){//catch and send error to console
47. console.error(e);
48. }
49. finally {// to close the connection
50. await client.close();
51. }
52. }
53. // call main function
54. main().catch(console.error);
55. // to create multi list
56. async function createMultipleListings(client, newListing){
57. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").insertMany(newListing);
58. console.log(`${result.insertedCount} New listing created with the following id (s):`);
59. console.log(result.insertedIds);
60. }
61. // -----------create single list-------
62. async function createListing(client, newListing){
63. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").insertOne(newListing);
64. console.log(`New listing created with the following id: ${result.insertedId}`)
65. }
66. //list database in cluster
67. async function listDatabases(client){
68. const databaseList = await client.db().admin().listDatabases(); //we have now list of databases
69. //to print them
70. console.log("Databases:");
71. databaseList.databases.forEach(db => {
73. console.log(`- ${db.name}`);
74. });
75. }

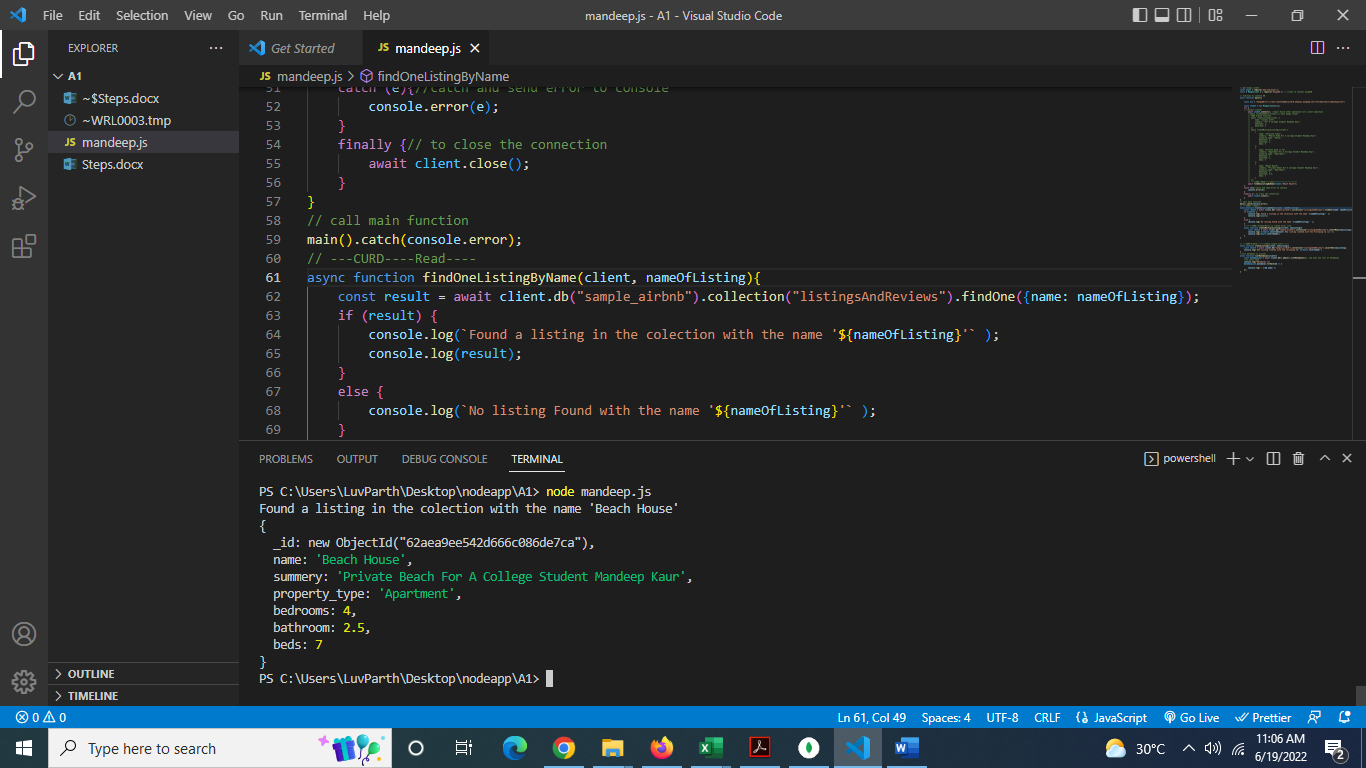


1. CURD Read Operation code

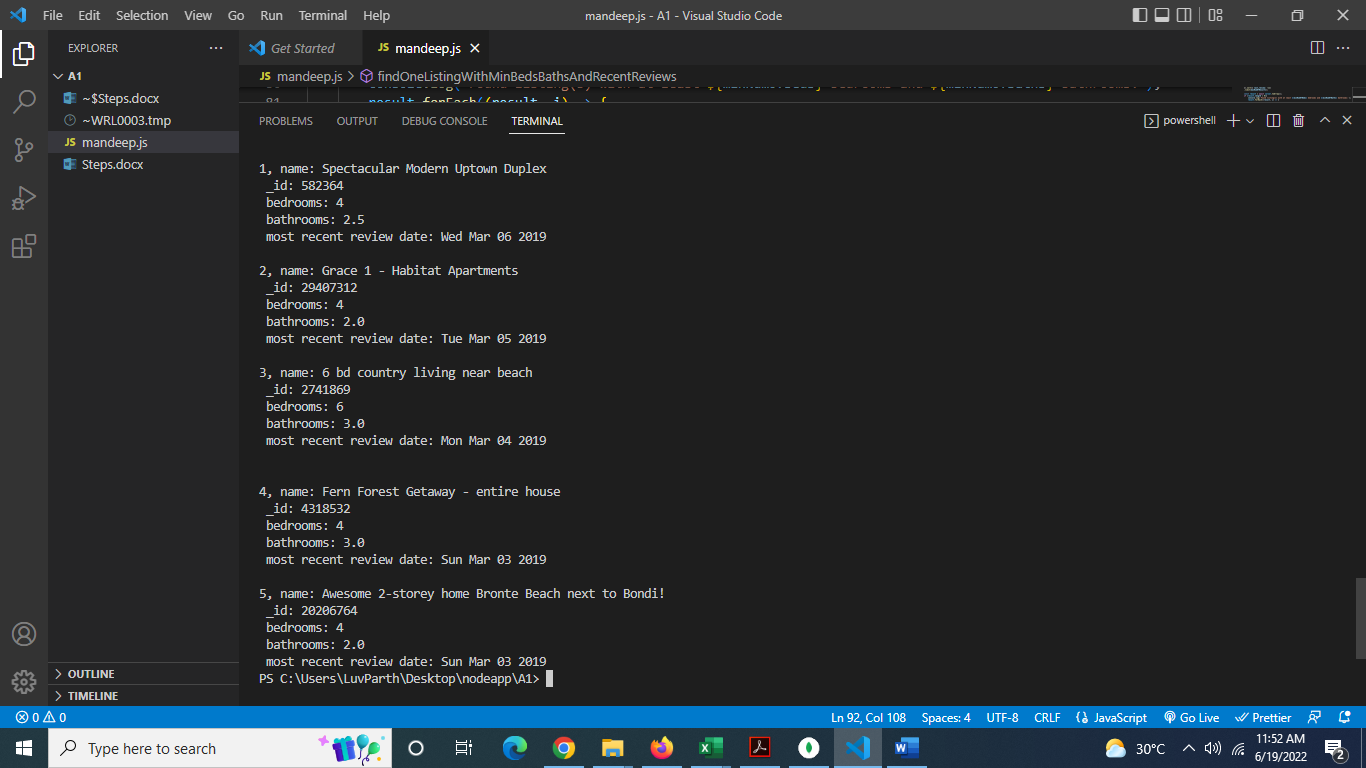
Read a record by passing a field’s value - e.g.,

FindOneListingByName()

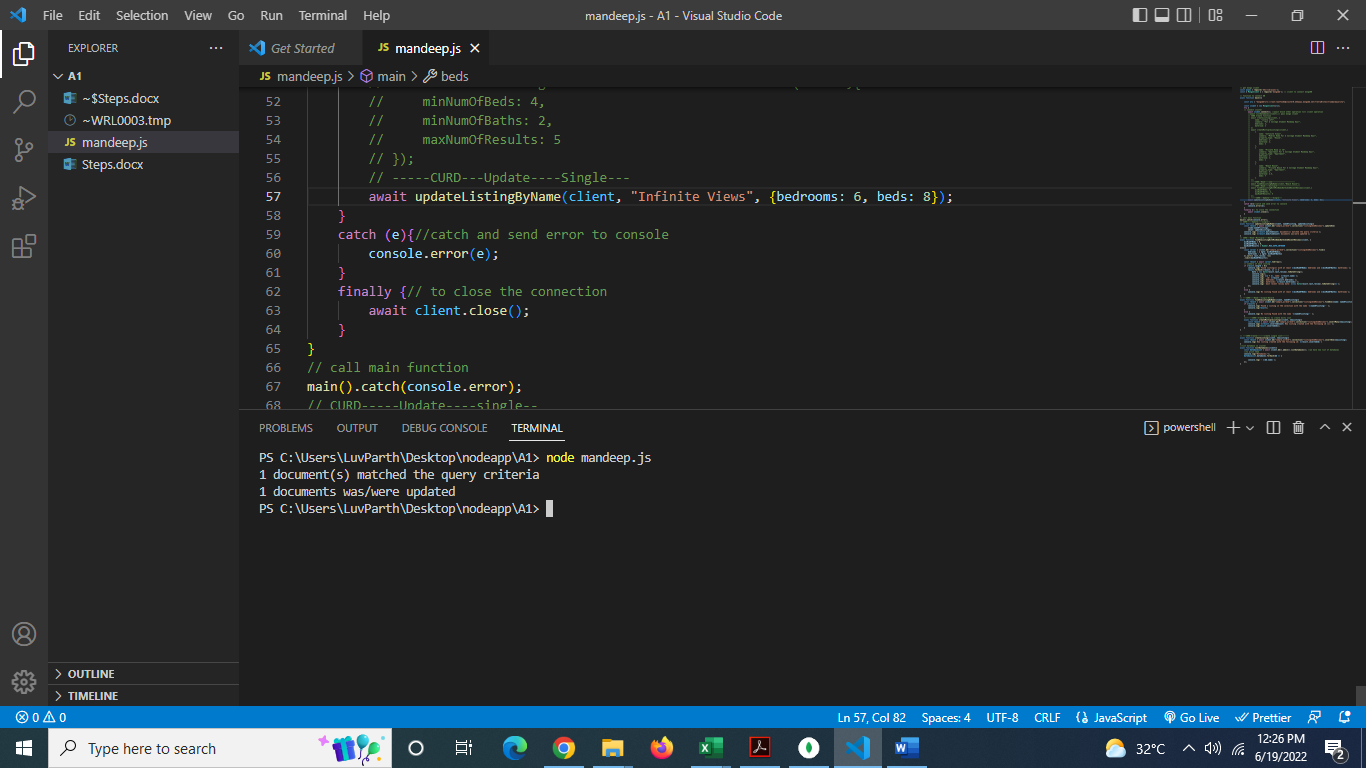
1. // get mongo client
2. const async = require('hbs/lib/async');
3. const { MongoClient } = require('mongodb'); // client to connect mongoDB
4. // function to connect DB
5. async function main(){
6. const uri = "mongodb+srv://root:PASSWORD@cluster0.ehkayyu.mongodb.net/?retryWrites=true&w=majority";
7. const client = new MongoClient(uri);
8. try {
9. // connect cluster
10. await client.connect(); //await block other operation till client operation
11. //await listDatabases(client);// pass mongo client
12. // CURD Create function
13. // await createListing(client, {
14. //     name: "Lovely Loft",
15. //     summery: "For A College Student Mandeep Kaur",
16. //     bedrooms: 2,
17. //     bathroom: 2
18. // })
19. // await createMultipleListings(client,[
20. //     {
21. //         name: "Infinite Views",
22. //         summery: "Modren Home For A College Student Mandeep Kaur",
23. //         property\_type: "House",
24. //         bedrooms: 2,
25. //         bathroom: 2,
26. //         beds: 2
27. //     },
28. //     {
29. //         name: "Private Room in CA",
30. //         summery: "Apartment For A College Student Mandeep Kaur",
31. //         property\_type: "Apartment",
32. //         bedrooms: 1,
33. //         bathroom: 1,
34. //         beds: 1
35. //     },
36. //     {
37. //         name: "Beach House",
38. //         summery: "Private Beach For A College Student Mandeep Kaur",
39. //         property\_type: "Apartment",
40. //         bedrooms: 4,
41. //         bathroom: 2.5,
42. //         beds: 7
43. //     }
44. // ]);
45. // ----CURD--Read-----call-----------------------
46. await findOneListingByName(client,"Beach House");
47. }
48. catch (e){//catch and send error to console
49. console.error(e);
50. }
51. finally {// to close the connection
52. await client.close();
53. }
54. }
55. // call main function
56. main().catch(console.error);
57. // ---CURD----Read----
58. async function findOneListingByName(client, nameOfListing){
59. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").findOne({name: nameOfListing});
60. if (result) {
61. console.log(`Found a listing in the colection with the name '${nameOfListing}'` );
62. console.log(result);
63. }
64. else {
65. console.log(`No listing Found with the name '${nameOfListing}'` );
66. }
67. //------CURD--Create-Multi to create multi list
68. async function createMultipleListings(client, newListing){
69. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").insertMany(newListing);
70. console.log(`${result.insertedCount} New listing created with the following id (s):`);
71. console.log(result.insertedIds);
72. }
73. }
74. // ---CURD-Create-------create single list-------
75. async function createListing(client, newListing){
76. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").insertOne(newListing);
77. console.log(`New listing created with the following id: ${result.insertedId}`)
78. }
79. //list database in cluster
80. async function listDatabases(client){
81. const databaseList = await client.db().admin().listDatabases(); //we have now list of databases
82. //to print them
83. console.log("Databases:");
84. databaseList.databases.forEach(db => {
86. console.log(`- ${db.name}`);
87. });
88. }



1. CURD for multiple read operation code
2. // get mongo client
3. const async = require('hbs/lib/async');
4. const { MongoClient } = require('mongodb'); // client to connect mongoDB
5. // function to connect DB
6. async function main(){
7. const uri = "mongodb+srv://root:PASSWORD@cluster0.ehkayyu.mongodb.net/?retryWrites=true&w=majority";
8. const client = new MongoClient(uri);
9. try {
10. // connect cluster
11. await client.connect(); //await block other operation till client operation
12. //await listDatabases(client);// pass mongo client
13. // CURD Create function
14. // await createListing(client, {
15. //     name: "Lovely Loft",
16. //     summery: "For A College Student Mandeep Kaur",
17. //     bedrooms: 2,
18. //     bathroom: 2
19. // })
20. // await createMultipleListings(client,[
21. //     {
22. //         name: "Infinite Views",
23. //         summery: "Modren Home For A College Student Mandeep Kaur",
24. //         property\_type: "House",
25. //         bedrooms: 2,
26. //         bathroom: 2,
27. //         beds: 2
28. //     },
29. //     {
30. //         name: "Private Room in CA",
31. //         summery: "Apartment For A College Student Mandeep Kaur",
32. //         property\_type: "Apartment",
33. //         bedrooms: 1,
34. //         bathroom: 1,
35. //         beds: 1
36. //     },
37. //     {
38. //         name: "Beach House",
39. //         summery: "Private Beach For A College Student Mandeep Kaur",
40. //         property\_type: "Apartment",
41. //         bedrooms: 4,
42. //         bathroom: 2.5,
43. //         beds: 7
44. //     }
45. // ]);
46. // ----CURD--Read----one-----------------------
47. // await findOneListingByName(client,"Beach House");
48. // ----CURD--Read----multiple-----------------------
49. await findOneListingWithMinBedsBathsAndRecentReviews(client,{
50. minNumOfBeds: 4,
51. minNumOfBaths: 2,
52. maxNumOfResults: 5
53. });
54. }
55. catch (e){//catch and send error to console
56. console.error(e);
57. }
58. finally {// to close the connection
59. await client.close();
60. }
61. }
62. // call main function
63. main().catch(console.error);
64. // CURD---Read--Multiple----Data--------
65. async function findOneListingWithMinBedsBathsAndRecentReviews(client, {
66. minNumOfBeds = 0,
67. minNumOfBaths = 0,
68. maxNumOfResults = Number.MAX\_SAFE\_INTEGER
69. }={}){
70. const cursor = client.db("sample\_airbnb").collection("listingsAndReviews").find({
71. bedrooms : { $gte: minNumOfBeds},
72. bathrooms : { $gte: minNumOfBaths}
73. }).sort({ last\_review: -1})
74. .limit(maxNumOfResults);
75. const result = await cursor.toArray();
76. if (result.length > 0){
77. console.log(`Found listing(s) with at least ${minNumOfBeds} bedrooms and ${minNumOfBaths} bathrooms:`);
78. result.forEach((result, i) => {
79. date = new Date(result.last\_review).toDateString();
80. console.log();
81. console.log(`${i + 1}, name: ${result.name}`);
82. console.log(` \_id: ${result.\_id}`);
83. console.log(` bedrooms: ${result.bedrooms}`);
84. console.log(` bathrooms: ${result.bathrooms}`);
85. console.log(` most recent review date: ${new Date(result.last\_review).toDateString()}`);
86. });
87. }
88. else {
89. console.log(`No listing found with at least ${minNumOfBeds} bedrooms and ${minNumOfBaths} bathrooms`);
90. }
91. }
92. // ---CURD----Read---Single-Record
93. async function findOneListingByName(client, nameOfListing){
94. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").findOne({name: nameOfListing});
95. if (result) {
96. console.log(`Found a listing in the colection with the name '${nameOfListing}'` );
97. console.log(result);
98. }
99. else {
100. console.log(`No listing Found with the name '${nameOfListing}'` );
101. }
102. //------CURD--Create-Multi to create multi list
103. async function createMultipleListings(client, newListing){
104. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").insertMany(newListing);
105. console.log(`${result.insertedCount} New listing created with the following id (s):`);
106. console.log(result.insertedIds);
107. }
108. }
109. // ---CURD-Create-------create single list-------
110. async function createListing(client, newListing){
111. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").insertOne(newListing);
112. console.log(`New listing created with the following id: ${result.insertedId}`)
113. }
114. //list database in cluster
115. async function listDatabases(client){
116. const databaseList = await client.db().admin().listDatabases(); //we have now list of databases
117. //to print them
118. console.log("Databases:");
119. databaseList.databases.forEach(db => {
121. console.log(`- ${db.name}`);
122. });
123. }



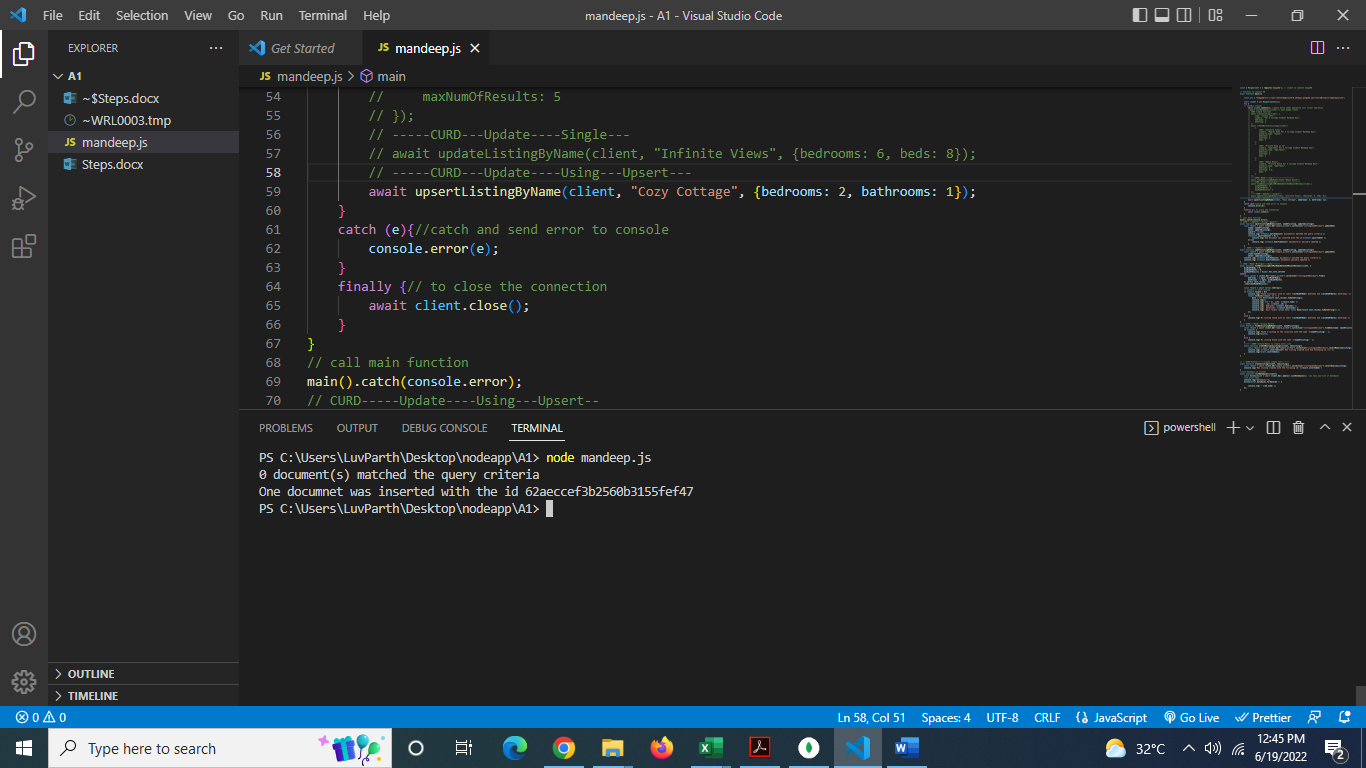
1. CURD update code
2. // get mongo client
3. const async = require('hbs/lib/async');
4. const { MongoClient } = require('mongodb'); // client to connect mongoDB
5. // function to connect DB
6. async function main(){
7. const uri = "mongodb+srv://root:PASSWORD@cluster0.ehkayyu.mongodb.net/?retryWrites=true&w=majority";
8. const client = new MongoClient(uri);
9. try {
10. // connect cluster
11. await client.connect(); //await block other operation till client operation
12. //await listDatabases(client);// pass mongo client
13. // CURD Create function
14. // await createListing(client, {
15. //     name: "Lovely Loft",
16. //     summery: "For A College Student Mandeep Kaur",
17. //     bedrooms: 2,
18. //     bathroom: 2
19. // })
20. // await createMultipleListings(client,[
21. //     {
22. //         name: "Infinite Views",
23. //         summery: "Modren Home For A College Student Mandeep Kaur",
24. //         property\_type: "House",
25. //         bedrooms: 2,
26. //         bathroom: 2,
27. //         beds: 2
28. //     },
29. //     {
30. //         name: "Private Room in CA",
31. //         summery: "Apartment For A College Student Mandeep Kaur",
32. //         property\_type: "Apartment",
33. //         bedrooms: 1,
34. //         bathroom: 1,
35. //         beds: 1
36. //     },
37. //     {
38. //         name: "Beach House",
39. //         summery: "Private Beach For A College Student Mandeep Kaur",
40. //         property\_type: "Apartment",
41. //         bedrooms: 4,
42. //         bathroom: 2.5,
43. //         beds: 7
44. //     }
45. // ]);
46. // ----CURD--Read----one-----------------------
47. // await findOneListingByName(client,"Beach House");
48. // ----CURD--Read----multiple-----------------------
49. // await findOneListingWithMinBedsBathsAndRecentReviews(client,{
50. //     minNumOfBeds: 4,
51. //     minNumOfBaths: 2,
52. //     maxNumOfResults: 5
53. // });
54. // -----CURD---Update----Single---
55. await updateListingByName(client, "Infinite Views", {bedrooms: 6, beds: 8});
56. }
57. catch (e){//catch and send error to console
58. console.error(e);
59. }
60. finally {// to close the connection
61. await client.close();
62. }
63. }
64. // call main function
65. main().catch(console.error);
66. // CURD-----Update----single--
67. async function updateListingByName(client, nameOfListing, updatedListing){
68. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").updateOne(
69. {name:nameOfListing},
70. {$set: updatedListing});
71. console.log(`${result.matchedCount} document(s) matched the query criteria`);
72. console.log(`${result.modifiedCount} documents was/were updated`);
73. }
74. // CURD---Read--Multiple----Data--------
75. async function findOneListingWithMinBedsBathsAndRecentReviews(client, {
76. minNumOfBeds = 0,
77. minNumOfBaths = 0,
78. maxNumOfResults = Number.MAX\_SAFE\_INTEGER
79. }={}){
80. const cursor = client.db("sample\_airbnb").collection("listingsAndReviews").find({
81. bedrooms : { $gte: minNumOfBeds},
82. bathrooms : { $gte: minNumOfBaths}
83. }).sort({ last\_review: -1})
84. .limit(maxNumOfResults);
85. const result = await cursor.toArray();
86. // summary of each listing
87. if (result.length > 0){
88. console.log(`Found listing(s) with at least ${minNumOfBeds} bedrooms and ${minNumOfBaths} bathrooms:`);
89. result.forEach((result, i) => {
90. date = new Date(result.last\_review).toDateString();
91. console.log();
92. console.log(`${i + 1}, name: ${result.name}`);
93. console.log(` \_id: ${result.\_id}`);
94. console.log(` bedrooms: ${result.bedrooms}`);
95. console.log(` bathrooms: ${result.bathrooms}`);
96. console.log(` most recent review date: ${new Date(result.last\_review).toDateString()}`);
97. });
98. }
99. else {
100. console.log(`No listing found with at least ${minNumOfBeds} bedrooms and ${minNumOfBaths} bathrooms`);
101. }
102. }
103. // ---CURD----Read---Single-Record
104. async function findOneListingByName(client, nameOfListing){
105. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").findOne({name: nameOfListing});
106. if (result) {
107. console.log(`Found a listing in the colection with the name '${nameOfListing}'` );
108. console.log(result);
109. }
110. else {
111. console.log(`No listing Found with the name '${nameOfListing}'` );
112. }
113. //------CURD--Create-Multi to create multi list
114. async function createMultipleListings(client, newListing){
115. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").insertMany(newListing);
116. console.log(`${result.insertedCount} New listing created with the following id (s):`);
117. console.log(result.insertedIds);
118. }
119. }
120. // ---CURD-Create-------create single list-------
121. async function createListing(client, newListing){
122. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").insertOne(newListing);
123. console.log(`New listing created with the following id: ${result.insertedId}`)
124. }
125. //list database in cluster
126. async function listDatabases(client){
127. const databaseList = await client.db().admin().listDatabases(); //we have now list of databases
128. //to print them
129. console.log("Databases:");
130. databaseList.databases.forEach(db => {
132. console.log(`- ${db.name}`);
133. });
134. }



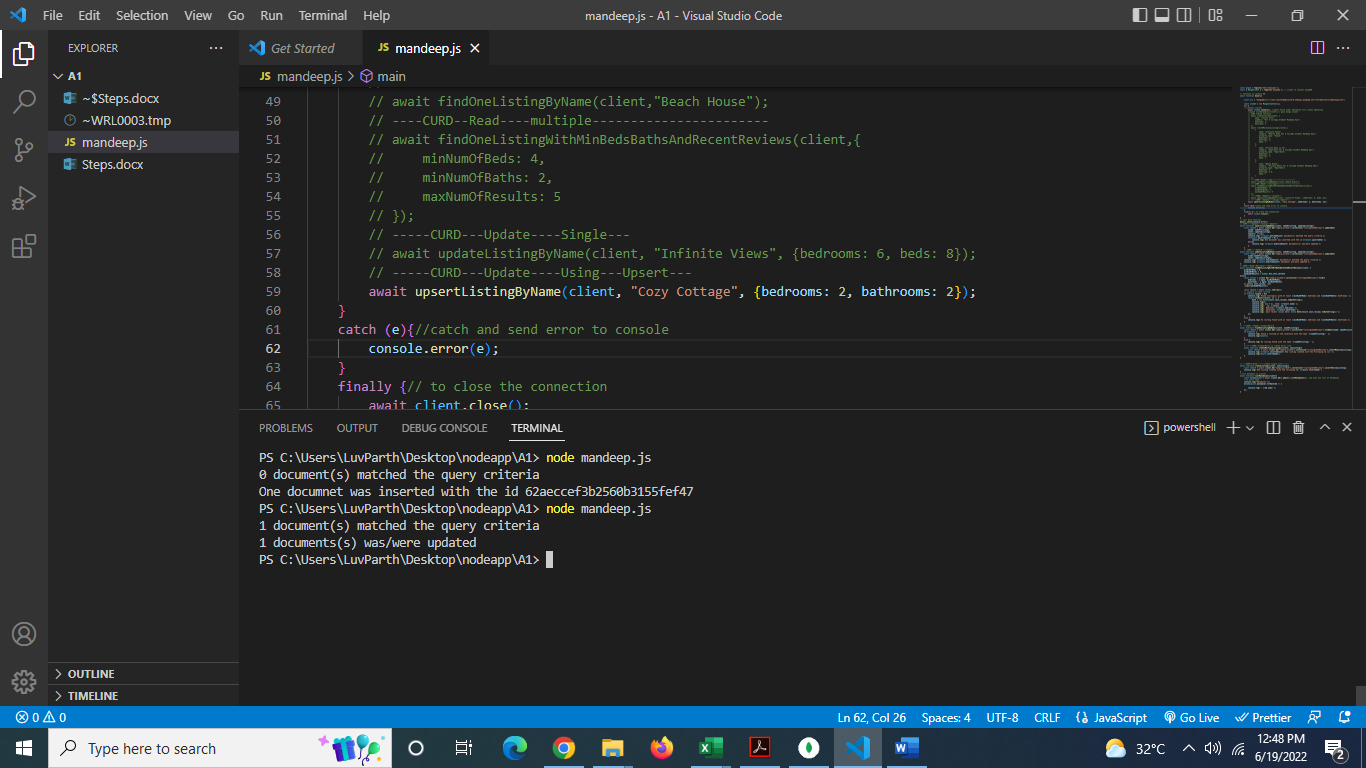
1. CURD Update using Upsert

Update an existing record using the upsert functionality

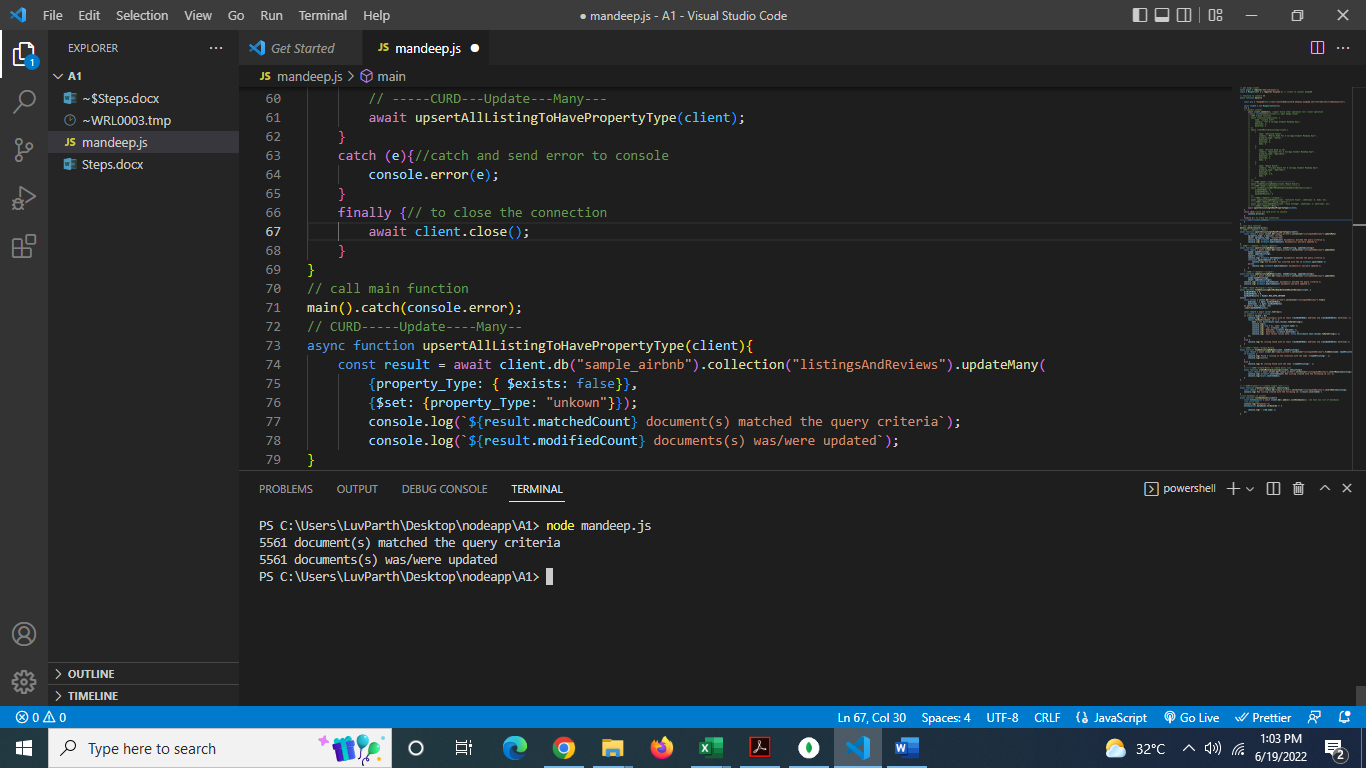
1. // get mongo client
2. const async = require('hbs/lib/async');
3. const { MongoClient } = require('mongodb'); // client to connect mongoDB
4. // function to connect DB
5. async function main(){
6. const uri = "mongodb+srv://root:PASSWORD@cluster0.ehkayyu.mongodb.net/?retryWrites=true&w=majority";
7. const client = new MongoClient(uri);
8. try {
9. // connect cluster
10. await client.connect(); //await block other operation till client operation
11. //await listDatabases(client);// pass mongo client
12. // CURD Create function
13. // await createListing(client, {
14. //     name: "Lovely Loft",
15. //     summery: "For A College Student Mandeep Kaur",
16. //     bedrooms: 2,
17. //     bathroom: 2
18. // })
19. // await createMultipleListings(client,[
20. //     {
21. //         name: "Infinite Views",
22. //         summery: "Modren Home For A College Student Mandeep Kaur",
23. //         property\_type: "House",
24. //         bedrooms: 2,
25. //         bathroom: 2,
26. //         beds: 2
27. //     },
28. //     {
29. //         name: "Private Room in CA",
30. //         summery: "Apartment For A College Student Mandeep Kaur",
31. //         property\_type: "Apartment",
32. //         bedrooms: 1,
33. //         bathroom: 1,
34. //         beds: 1
35. //     },
36. //     {
37. //         name: "Beach House",
38. //         summery: "Private Beach For A College Student Mandeep Kaur",
39. //         property\_type: "Apartment",
40. //         bedrooms: 4,
41. //         bathroom: 2.5,
42. //         beds: 7
43. //     }
44. // ]);
45. // ----CURD--Read----one-----------------------
46. // await findOneListingByName(client,"Beach House");
47. // ----CURD--Read----multiple-----------------------
48. // await findOneListingWithMinBedsBathsAndRecentReviews(client,{
49. //     minNumOfBeds: 4,
50. //     minNumOfBaths: 2,
51. //     maxNumOfResults: 5
52. // });
53. // -----CURD---Update----Single---
54. // await updateListingByName(client, "Infinite Views", {bedrooms: 6, beds: 8});
55. // -----CURD---Update----Using---Upsert---
56. await upsertListingByName(client, "Cozy Cottage", {bedrooms: 2, bathrooms: 1});
57. }
58. catch (e){//catch and send error to console
59. console.error(e);
60. }
61. finally {// to close the connection
62. await client.close();
63. }
64. }
65. // call main function
66. main().catch(console.error);
67. // CURD-----Update----Using---Upsert--
68. async function upsertListingByName(client, nameOfListing, updatedListing){
69. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").updateOne(
70. {name: nameOfListing},
71. {$set: updatedListing},
72. {upsert: true});
73. console.log(`${result.matchedCount} document(s) matched the query criteria`);
74. if(result.upsertedCount > 0) {
75. console.log(`One documnet was inserted with the id ${result.upsertedId}`);
76. }else{
77. console.log(`${result.modifiedCount} documents(s) was/were updated`);
78. }
79. }
80. // CURD-----Update----single--
81. async function updateListingByName(client, nameOfListing, updatedListing){
82. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").updateOne(
83. {name:nameOfListing},
84. {$set: updatedListing});
85. console.log(`${result.matchedCount} document(s) matched the query criteria`);
86. console.log(`${result.modifiedCount} documents was/were updated`);
87. }
88. // CURD---Read--Multiple----Data--------
89. async function findOneListingWithMinBedsBathsAndRecentReviews(client, {
90. minNumOfBeds = 0,
91. minNumOfBaths = 0,
92. maxNumOfResults = Number.MAX\_SAFE\_INTEGER
93. }={}){
94. const cursor = client.db("sample\_airbnb").collection("listingsAndReviews").find({
95. bedrooms : { $gte: minNumOfBeds},
96. bathrooms : { $gte: minNumOfBaths}
97. }).sort({ last\_review: -1})
98. .limit(maxNumOfResults);
99. const result = await cursor.toArray();
100. // summary of each listing
101. if (result.length > 0){
102. console.log(`Found listing(s) with at least ${minNumOfBeds} bedrooms and ${minNumOfBaths} bathrooms:`);
103. result.forEach((result, i) => {
104. date = new Date(result.last\_review).toDateString();
105. console.log();
106. console.log(`${i + 1}, name: ${result.name}`);
107. console.log(` \_id: ${result.\_id}`);
108. console.log(` bedrooms: ${result.bedrooms}`);
109. console.log(` bathrooms: ${result.bathrooms}`);
110. console.log(` most recent review date: ${new Date(result.last\_review).toDateString()}`);
111. });
112. }
113. else {
114. console.log(`No listing found with at least ${minNumOfBeds} bedrooms and ${minNumOfBaths} bathrooms`);
115. }
116. }
117. // ---CURD----Read---Single-Record
118. async function findOneListingByName(client, nameOfListing){
119. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").findOne({name: nameOfListing});
120. if (result) {
121. console.log(`Found a listing in the colection with the name '${nameOfListing}'` );
122. console.log(result);
123. }
124. else {
125. console.log(`No listing Found with the name '${nameOfListing}'` );
126. }
127. //------CURD--Create-Multi to create multi list
128. async function createMultipleListings(client, newListing){
129. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").insertMany(newListing);
130. console.log(`${result.insertedCount} New listing created with the following id (s):`);
131. console.log(result.insertedIds);
132. }
133. }
134. // ---CURD-Create-------create single list-------
135. async function createListing(client, newListing){
136. const result = await client.db("sample\_airbnb").collection("listingsAndReviews").insertOne(newListing);
137. console.log(`New listing created with the following id: ${result.insertedId}`)
138. }
139. //list database in cluster
140. async function listDatabases(client){
141. const databaseList = await client.db().admin().listDatabases(); //we have now list of databases
142. //to print them
143. console.log("Databases:");
144. databaseList.databases.forEach(db => {
146. console.log(`- ${db.name}`);
147. });
148. }



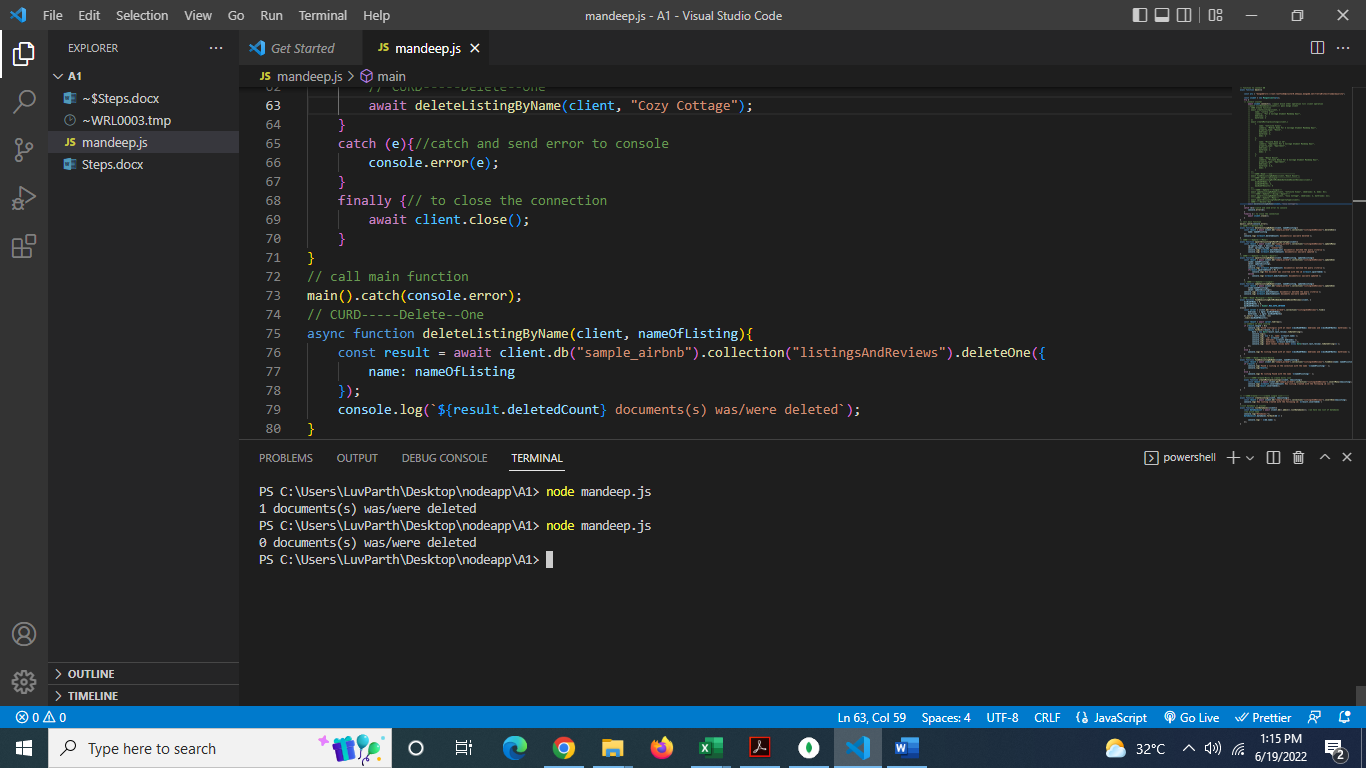
1. Same code with updated data



1. CURD Update Many code

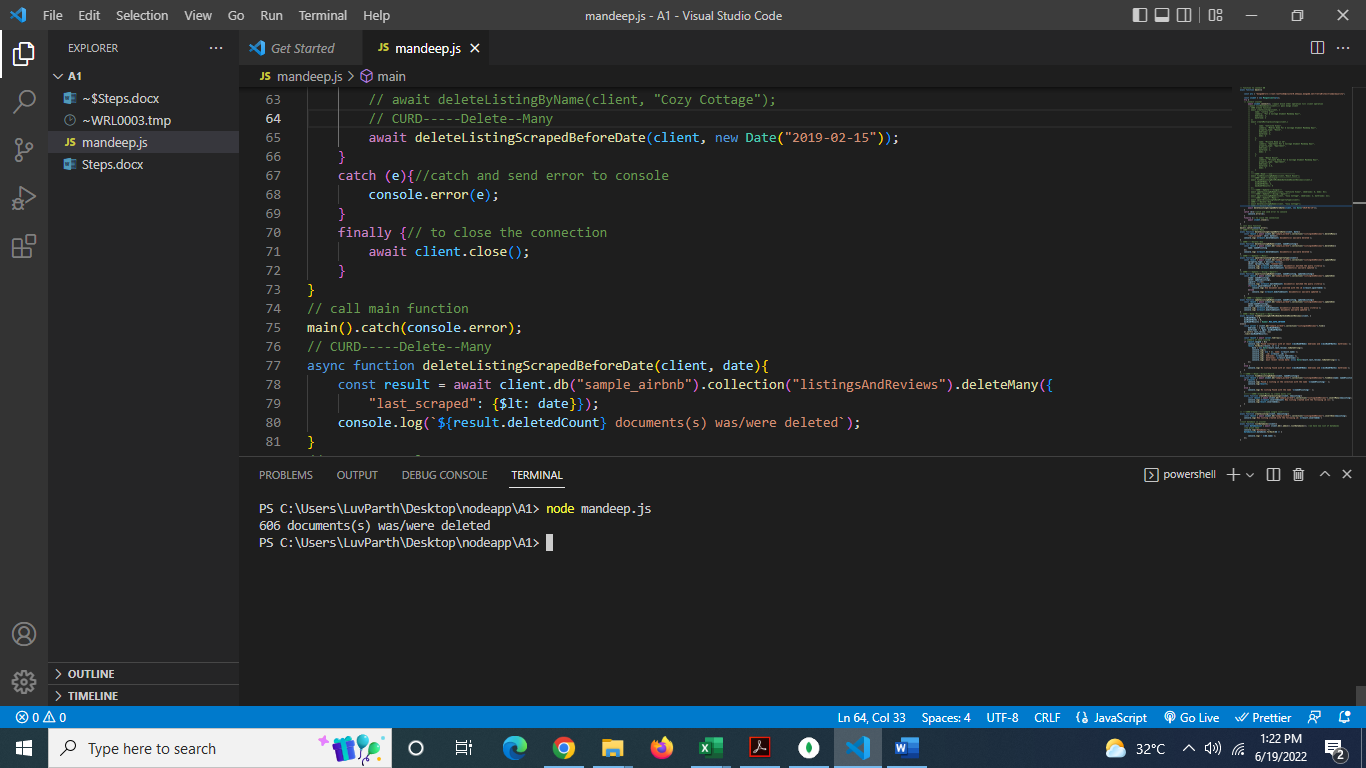


1. CURD Delete Code



1. CURD Delete Many

Delete records by date – e.g., deleteListingsScrapedBeforeDate()



**Sr.No: 4**

1. Reading records and presenting it on a web browser (running on loacalhost)

**CODE: server.js**

const express = require('express');

const mongoose = require('mongoose');

const app = express();

const ejs = require('ejs');

const { kStringMaxLength } = require('buffer');

app.set('view engine', 'ejs');

mongoose.connect('mongodb+srv://root:root71162@cluster0.ehkayyu.mongodb.net/SchoolManagementSystem?retryWrites=true&w=majority');

const studentsSchema = {

    StudentId: String,

    FirstName: String,

    LastName: String,

    Email: String,

    Mobile: String

}

const Student = mongoose.model('Student',studentsSchema);

app.get('/', (req, res) => {

    Student.find({}, function(err, Student) {

        res.render('index', {

            StudentsList: Student

        })

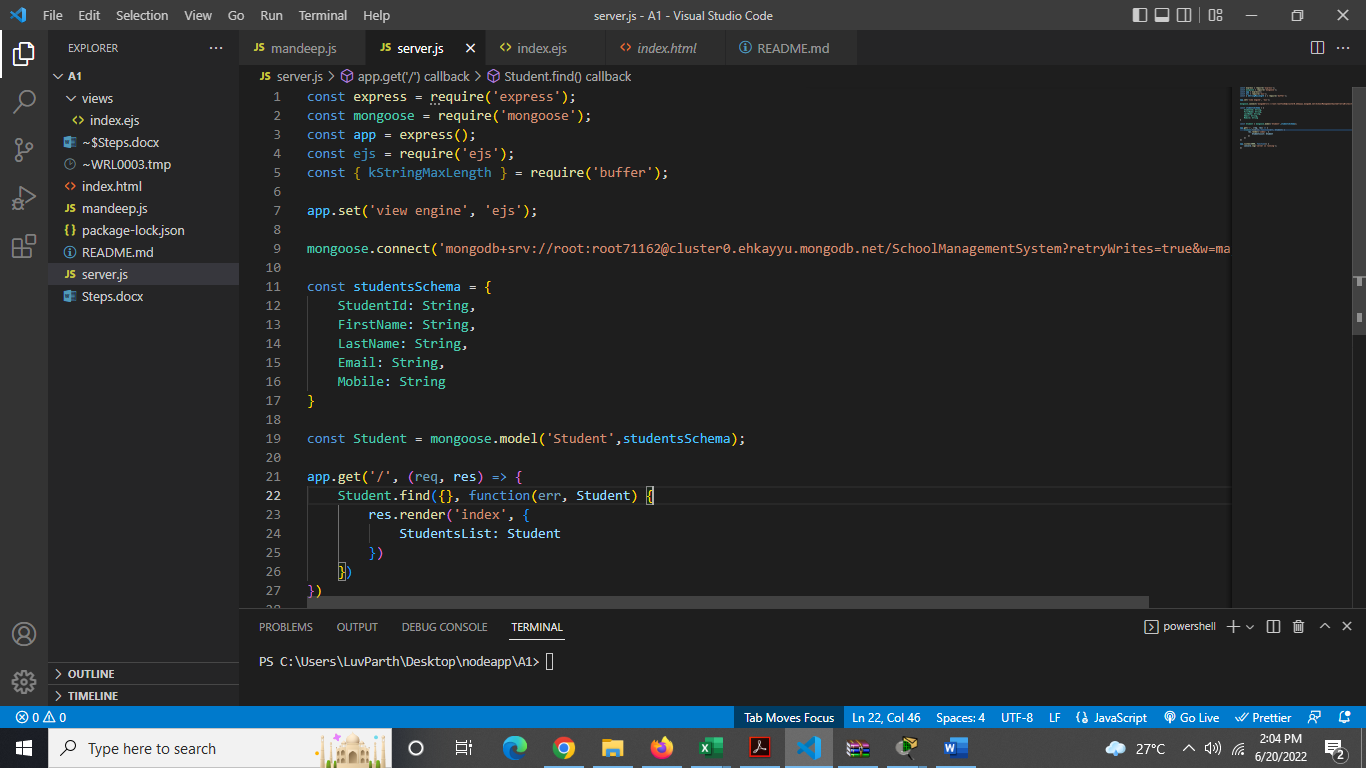
    })

})

app.listen(4000, function() {

    console.log('server is running');

})



**CODE: index.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

</head>

<body style="background-color:#f3f3f3";>

  <h1 style="text-align: center; text-transform: uppercase; color: #009879;">School Management System</h1>

    <table class="table" style="border-collapse: collapse; box-shadow: 0 0 20px rgba(0, 0, 0, 0.15);">

        <thead style="background-color: #009879; color: #ffffff;">

          <tr><th style="text-align:center; text-transform:uppercase ; " colspan="6">Student Data</th></tr>

          <tr>

            <th scope="col">UID</th>

            <th scope="col">Student Id</th>

            <th scope="col">First Name</th>

            <th scope="col">Last Name</th>

            <th scope="col">Email</th>

            <th scope="col">Mobile</th>

          </tr>

        </thead>

        <tbody style="background-color: #f3f3f3;">

        <%StudentsList.forEach(Student => {%>

            <tr>

                <th scope="row"><%= Student.\_id %></th>

                <td><%= Student.StudentId %></td>

                <td><%= Student.FirstName %></td>

                <td><%= Student.LastName %></td>

                <td><%= Student.Email %></td>

                <td><%= Student.Mobile %></td>

            </tr>

        <%})%>

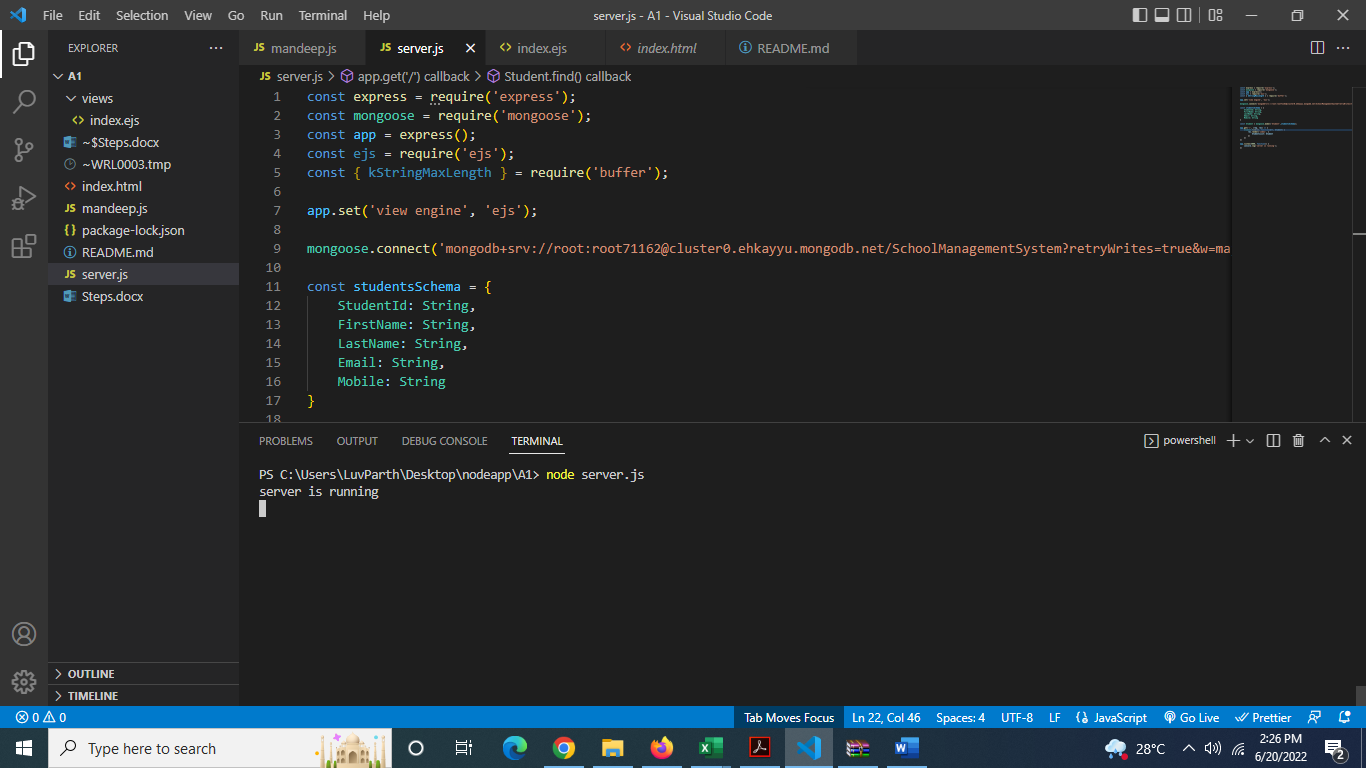
        </tbody>

      </table>

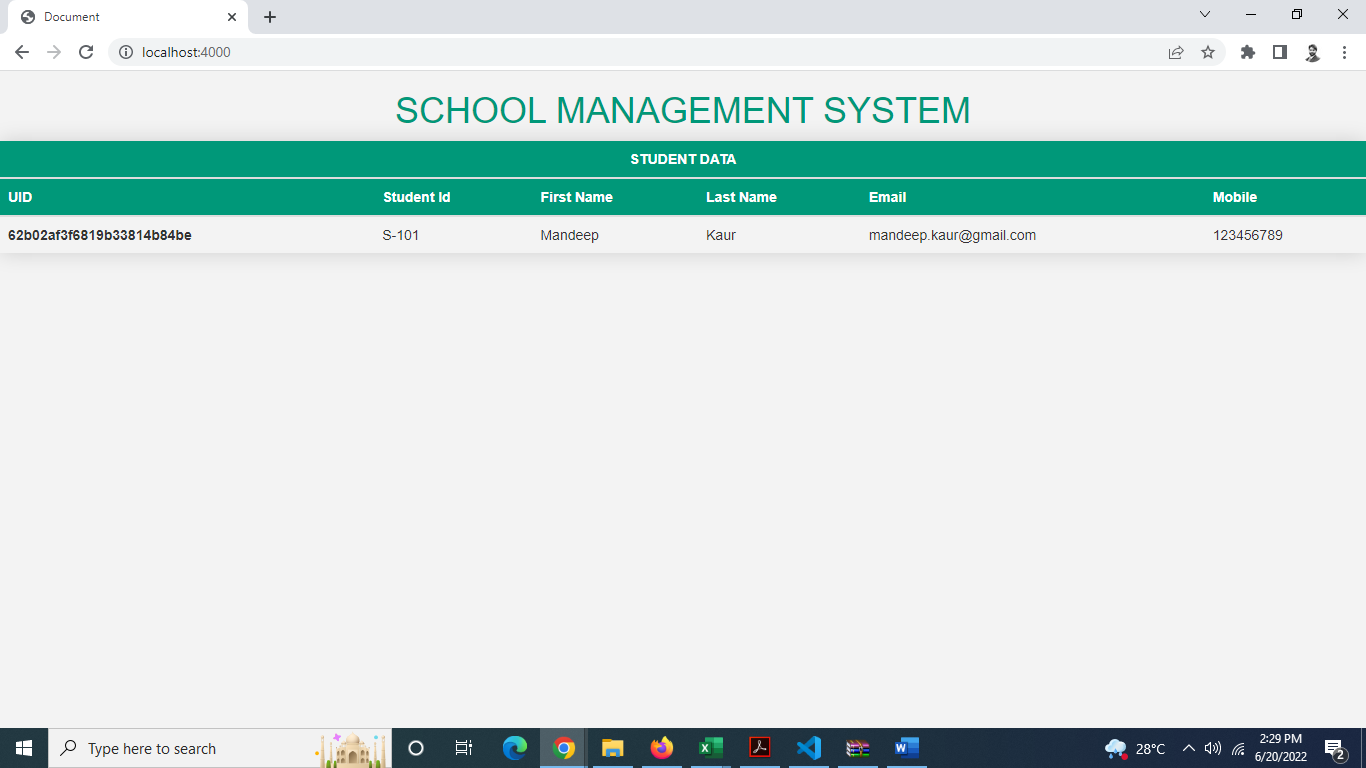
</body>

</html>

**Running Server: node server.js**



**Presenting it on a web browser (running on loacalhost:4000)**



README.md

