Lab 9 - MongoDB Relationship & Azure Cognitive Service

Please **download your Express.js application**.

Suppose for each new booking, we generate the **number of tickets** as specified by the user and each ticket has its own **QR code which embeds an UUID**.

To work with uuid, let's install the dependency:

```
npm install uuid --save
```

In `index.js`, add

```
const { v4: uuidv4 } = require('uuid');
```

Create Tickets for a Booking

Modify the booking **create route handler** as follows:

```
/* Handle the Form */
router.post('/bookings', async function (req, res) {

  req.body.numTickets = parseInt(req.body.numTickets);

  let result = await db.collection("bookings").insertOne(req.body);

  for (var i = 0; i < req.body.numTickets; i++) {

    await db.collection("tickets").insertOne({ bookingId:
result.insertedId, uuid: uuidv4() });
  }

  res.status(201).json({ id: result.insertedId });

});
```

Try adding a new booking, and take a look at the database, you should find a new collection `tickets` with some documents. Each document contains **a random** `uuid`, its **primary key** `_id`, and most importantly, the `bookingId`, which is the primary key of the booking.

Retrieve the Tickets of a Booking

Let's implement a new route handler which extracts the tickets of each booking.

```
router.get("/api/bookings/:id/tickets", async function (req, res) {

  if (!ObjectId.isValid(req.params.id))
    return res.status(404).send('Unable to find the requested resource!');

  var pipelines = [
    { $match: { _id: req.params.id } },
    {
      $lookup:
      {
        from: "tickets",
        localField: "_id",
        foreignField: "bookingId",
        as: "tickets"
      }
    }
  ]

  let results = await
db.collection("bookings").aggregate(pipelines).toArray();

  if (results.length > 0)
    return res.json(results[0]);
  else
    return res.status(404).send("Not Found");

});
```

Here, the use of `$lookup` will retrieve the corresponding tickets in the `tickets` collection. The tickets are now stored in the `tickets` property as specified with `as` above.

```
{
  "_id": "637504ba121cea706882ef4e",
  "email": "jdkfl@jdl.com",
  "numTickets": 2,
  "team": "Avengers",
  "superhero": "Thor",
  "payment": "Credit Card",
  "terms": "on",
  "tickets": [
    {
      "_id": "637504bb121cea706882ef4f",
      "bookingId": "637504ba121cea706882ef4e",
      "uuid": "c3293085-2b91-4588-96ab-86a5189ff533"
    },
    {
      "_id": "637504bb121cea706882ef50",
      "bookingId": "637504ba121cea706882ef4e",
      "uuid": "4c2c5503-f5a4-453c-bafd-d0b37ae0325b"
    }
  ]
}
```


Booking Form

Revisit the booking form `/index.html` and add a text area there.

```
<div class="form-floating">
  <textarea class="form-control" placeholder="Leave a comment here"
id="floatingTextarea" name="comment"></textarea>
  <label for="floatingTextarea">Comments</label>
</div>
```

Azure Cognitive Models

Login to the Azure portal and search for `Cognitive Services`, create a `Language Service`.

Hit `Continue to create resource`. Make sure you use the `F0` free tier.

Under `Resource Management`, you will need the `Keys and Endpoint` later.

Back to Visual Studio Code, in the terminal, run

```
npm install @azure/ai-text-analytics --save
```

Then, implement the following in `index.js`:

```
"use strict";

const { TextAnalyticsClient, AzureKeyCredential } = require("@azure/ai-text-analytics");
const key = '<paste-your-key-here>';
const endpoint = '<paste-your-endpoint-here>';
// Authenticate the client with your key and endpoint
const textAnalyticsClient = new TextAnalyticsClient(endpoint,  new AzureKeyCredential(key));

// Example method for detecting sentiment in text
async function sentimentAnalysis(client){

    const sentimentInput = [
        "I had the best day of my life. I wish you were there with me."
    ];
    const sentimentResult = await client.analyzeSentiment(sentimentInput);

    sentimentResult.forEach(document => {
        console.log(`ID: ${document.id}`);
        console.log(`\tDocument Sentiment: ${document.sentiment}`);
        console.log(`\tDocument Scores:`);
        console.log(`\t\tPositive: ${document.confidenceScores.positive.toFixed(2)} \tNegative: ${document.confidenceScores.negative.toFixed(2)} \tNeutral: ${document.confidenceScores.neutral.toFixed(2)}`);
        console.log(`\tSentences Sentiment(${document.sentences.length}):`);
        document.sentences.forEach(sentence => {
            console.log(`\t\tSentence sentiment: ${sentence.sentiment}`)
            console.log(`\t\tSentences Scores:`);
```

```
            console.log(`\t\tPositive:
${sentence.confidenceScores.positive.toFixed(2)} \tNegative:
${sentence.confidenceScores.negative.toFixed(2)} \tNeutral:
${sentence.confidenceScores.neutral.toFixed(2)}`);
        });
    });
}
```

Reference: https://docs.microsoft.com/en-us/azure/cognitive-services/language-service/sentiment-opinion-mining/quickstart?pivots=programming-language-javascript

Paste the **key** and **endpoint** to the code above.

The Comment

Modify the function a bit as follows:

```
async function sentimentAnalysis(client, comment){

  const sentimentInput = [
      // "I had the best day of my life. I wish you were there with me."
      comment
  ];
  const sentimentResult = await client.analyzeSentiment(sentimentInput);

  sentimentResult.forEach(document => {
      console.log(`ID: ${document.id}`);
      console.log(`\tDocument Sentiment: ${document.sentiment}`);
      console.log(`\tDocument Scores:`);
      console.log(`\t\tPositive:
${document.confidenceScores.positive.toFixed(2)} \tNegative:
${document.confidenceScores.negative.toFixed(2)} \tNeutral:
${document.confidenceScores.neutral.toFixed(2)}`);
      console.log(`\tSentences Sentiment(${document.sentences.length}):`);
      document.sentences.forEach(sentence => {
          console.log(`\t\tSentence sentiment: ${sentence.sentiment}`)
          console.log(`\t\tSentences Scores:`);
          console.log(`\t\tPositive:
```

```
${sentence.confidenceScores.positive.toFixed(2)} \tNegative:
${sentence.confidenceScores.negative.toFixed(2)} \tNeutral:
${sentence.confidenceScores.neutral.toFixed(2)}`);
      });
  });

  return sentimentResult;
}
```

Thus, the function **accepts the comment as the second argument**, and returns the **sentiment analysis result**.

Finally, modify the `router.post('/bookings')` route handler as follows:

```
/* Handle the Form submission with Restful Api */
router.post('/bookings', async function (req, res) {

  req.body.numTickets = parseInt(req.body.numTickets);

  req.body.analysis = await sentimentAnalysis(textAnalyticsClient,
req.body.comment);

    for (var i = 0; i < req.body.numTickets; i++) {

    await db.collection("tickets").insertOne({ bookingId:
result.insertedId, uuid: uuidv4() });
  }

  let result = await db.collection("bookings").insertOne(req.body);

  res.status(201).json({ id: result.insertedId });
});
```

The **sentiment analysis result** of the input message is now saved together with the submitted booking record.