Module 4 Activity 3: Creating a NoSQL Database in MongoDB Atlas

Ryan M.K. Burke-Jimenez

Arizona State University

IFT 458

Professor Dinesh Sthapit

October 8th, 2023

Contents

[Contents 2](#_Toc147441509)

[Module 4 Activity 3: Creating a NoSQL Database in MongoDB Atlas 4](#_Toc147441510)

[Initial Setup 4](#_Toc147441511)

[Create an Express Backend 5](#_Toc147441512)

[Step 1: Additional Environment Variables 5](#_Toc147441513)

[Config.env 5](#_Toc147441514)

[Step 2: Model Design for Loan Application 5](#_Toc147441515)

[Model Folder Creation 5](#_Toc147441516)

[loanModel.js 6](#_Toc147441517)

[Step 3: Server Connectivity – MongoDB Connection 7](#_Toc147441518)

[Step 4: Database Manager + Query Filtering 8](#_Toc147441519)

[Step 5: Controller for Loan Application 9](#_Toc147441520)

[Step 6: App Routes and Controllers Established 12](#_Toc147441521)

[Interact with MongoDB using Mongoose 12](#_Toc147441522)

[Test With Postman 14](#_Toc147441523)

[Add a New Loan 14](#_Toc147441524)

[Get All Loans 15](#_Toc147441525)

[Get a Specific Loan by ID 16](#_Toc147441526)

[Change an Existing Loan 16](#_Toc147441527)

[Delete a Loan 17](#_Toc147441528)

[GitHub Repo Link 19](#_Toc147441529)

# Module 4 Activity 3: Creating a NoSQL Database in MongoDB Atlas

This activity is a continuation from the first activity in this module where we created a NoSQL Database in MongoDB Atlas. We will be connecting to and experimenting with the cluster that was created. There are initial steps to the lab that must be completed before proceeding with the rest of the activity.

# Initial Setup

Before interacting with MongoDB via Mongoose, I installed the MongoDB Extension within VSCode to make connecting the cluster seamless:

*Screenshots of MongoDB extension installed*

A screenshot of a computer

Description automatically generated

In MongoDB Atlas, there is a token that the user may use to connect to the database within VSCode. It is also important to note that the user must have the MongoDB extension installed in VSCode as well. The MongoDB cluster was created in Module 4 Activity 1.

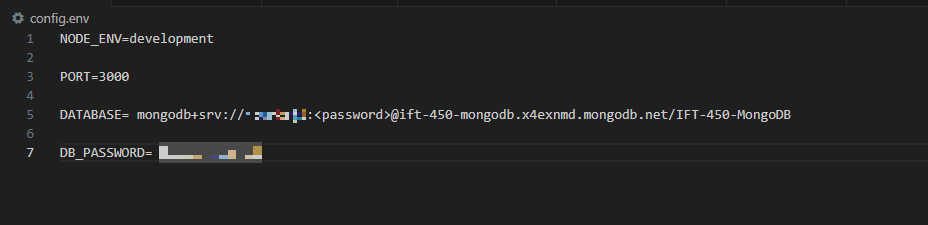
# Create an Express Backend

This section of the activity will focus on all files that are included in the loan application. Each pertinent file will have their own section within this area of the activity document with each section named after the steps within the lab.

## Step 1: Additional Environment Variables

### Config.env

Continuing from the previous section, we must create a config.env file that has vital information pertaining to the connectivity of the MongoDB cluster:

*Contents of config.env file*  


These details will be used within the main server.js file that will connect to the cloud database.

## Step 2: Model Design for Loan Application

### Model Folder Creation

Per the instructions of the activity, a ‘model’ folder was created to store three files: loanRoutes.js, loanController.js, and loanModel.js.

*Files/File Structure of Model Folder*

A screenshot of a program

Description automatically generated

### loanModel.js

The provided screenshots are descriptive of how the Loan model functions and what is expected within a query to the database:

A screenshot of a computer program

Description automatically generated*Screenshots of the Loan model structure*

A computer screen shot of a program

Description automatically generated

## Step 3: Server Connectivity – MongoDB Connection

In this section, the goal is to allow the application to successfully connect to the MongoDB database as well as being the listener for the application; this is accomplished within the server.js file. The config.env is used as to fill in information for the database connection:

*Screenshot of server.js file + MongoDB Connection*

A screen shot of a computer program

Description automatically generated

## Step 4: Database Manager + Query Filtering

In this section, a folder dubbed “dataBaseManager” is created and holds a file “loanDbContext.js”. LoandbContext.js is structured to supplement API features that manage query results:

*Screenshot of loanDbContext.js file + dataBaseManager folder created*

A screen shot of a computer program

Description automatically generated

## Step 5: Controller for Loan Application

The controller handles the logic and functionality of the application and provides try and catch architecture to handle errors and validate data entered. Each function represents requests that are standard in queries:

*Screenshots of loanController.js file*

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

## Step 6: App Routes and Controllers Established

The final file within the application is the app.js file. This will allow the application to be navigated within an internet browser and provide structure for the application:

*Screenshot of app.js file*

A screen shot of a computer program

Description automatically generated

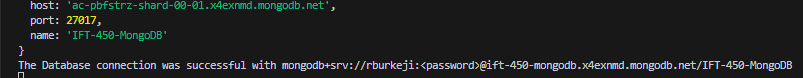
# Interact with MongoDB using Mongoose

As previously mentioned in Step 3, Mongoose is used to establish a connection to the MongoDB database. After files are properly implemented and established for the application, testing of the connection to the database is confirmed within the console of VSCode.

*Screenshots of successful connection to the MongoDB database*

*A screen shot of a computer

Description automatically generated*



*Screenshots of localhost:3000/api/v1/loans in Chrome*

*A screen shot of a computer

Description automatically generated*

*Screenshots of API in Postman*

*A black rectangular object with a white line

Description automatically generated*

# Test With Postman

The final part of this activity is to do some testing with Postman to confirm the API has full functionality. Each function described will have images of Postman and/or MongoDB.

## Add a New Loan

*Screenshots of adding (POST) a new loan (Postman 🡪 MongoDB)*

*A screenshot of a computer

Description automatically generated*

*A screenshot of a computer

Description automatically generated*

## Get All Loans

*Screenshot of Postman GET Request*

*A screenshot of a computer

Description automatically generated*

## Get a Specific Loan by ID

The ID used in this requirement is a unique ID that is provided by MongoDB. Within their system, MongoDB supplies a unique ID to act as the primary key within a MongoDB collection. This ID can be called in GET/PATCH/PUT/DELETE requests when necessary:

*Screenshot of Postman GET Request for a Specific Loan by ID*

*A screenshot of a computer

Description automatically generated*

## Change an Existing Loan

As previously mentioned, to enable the PATCH or PUT request for a change in an entry, we may use the ID of that entity to deploy the request.

*Screenshots of changing Bart’s Loan Amount from 4000 to 6500 (Postman 🡪 MongoDB)*

*A screenshot of a computer program

Description automatically generated*

*A screenshot of a computer

Description automatically generated*

## Delete a Loan

The final step in this activity is to demonstrate deleting a loan. Stated previously, we will use the unique ID to enact a deletion of Ash Ketchum’s loan account.

*Screenshots of deleting Ash’s Loan Account (Postman 🡪 MongoDB)*

*A screenshot of a computer

Description automatically generated*

*A screenshot of a computer

Description automatically generated*

# GitHub Repo Link

This activity has been uploaded to a GitHub Repository for further analysis:

<https://github.com/RyanBurkeJimenez/Module4Activity3>