Module 8: Assignment Final Project

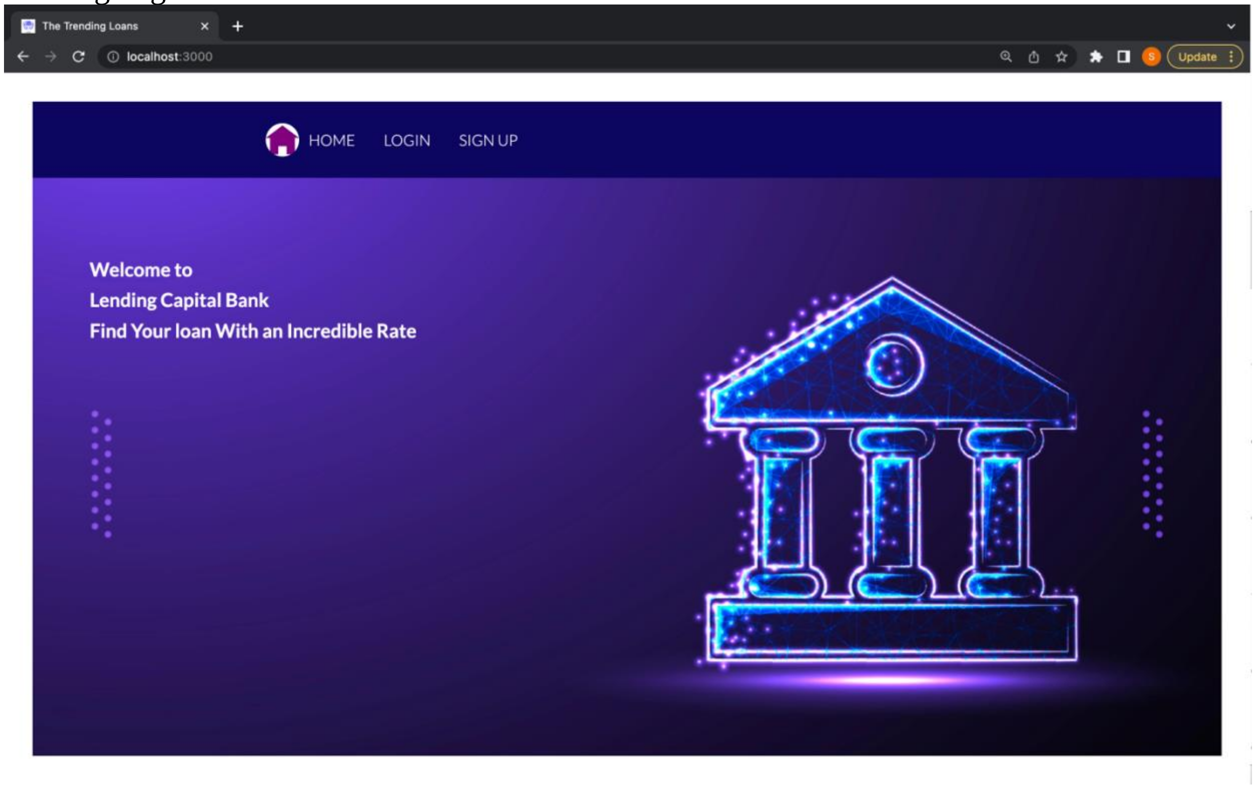
Spurthy Mutturaj

Instructor: Dinesh Sthapit

Dec 2,2022

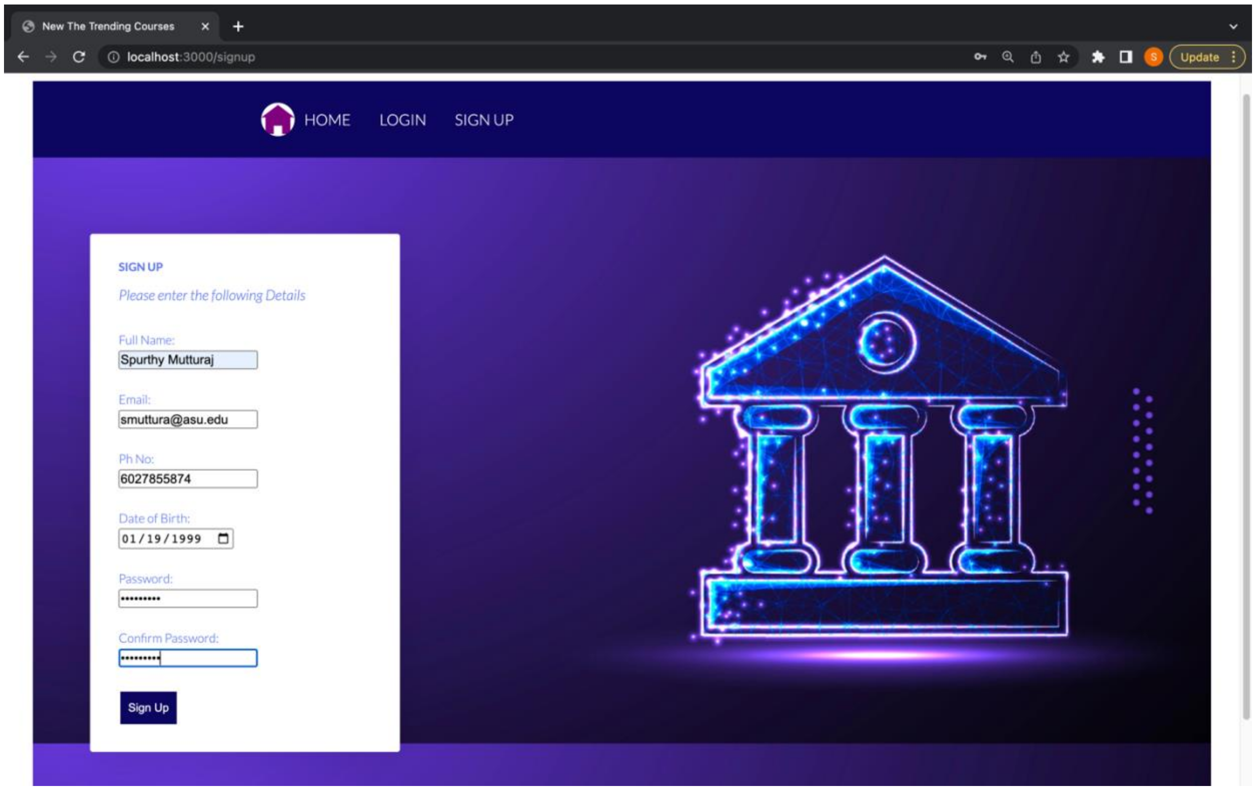
GitHub:

Landing Page



Welcome to
Lending Capital Bank
Find Your loan With an Incredible Rate

Sign Up page:



SIGN UP

Please enter the following Details

Full Name: Spurthy Mutturaj

Email: smuttura@asu.edu

Ph No: 6027855874

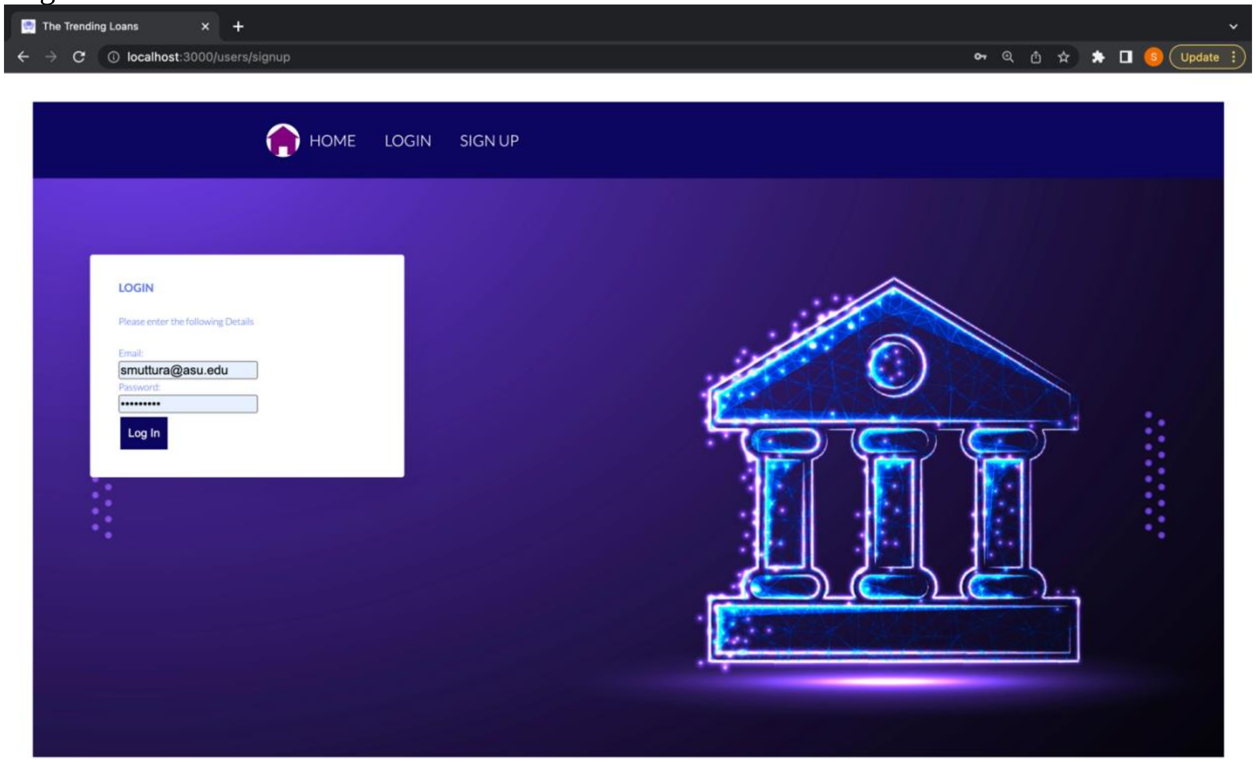
Date of Birth: 01/19/1999

Password: •••••••••

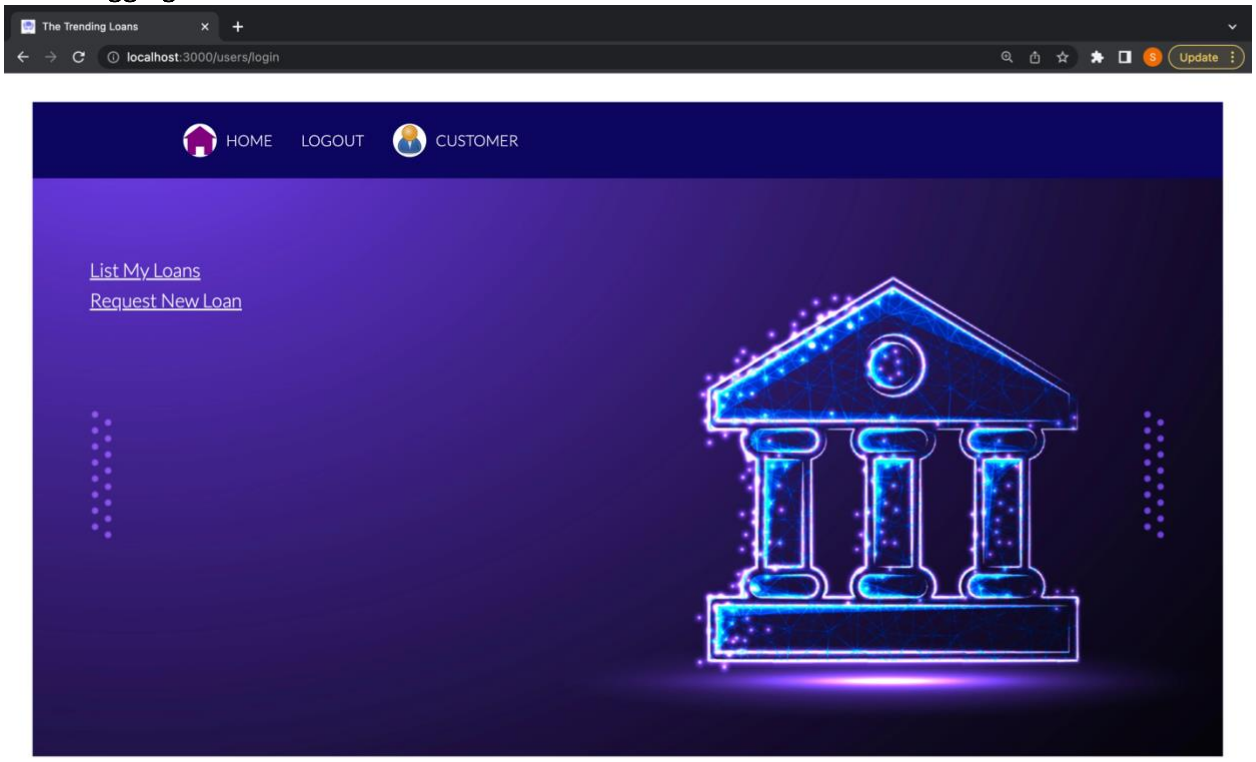
Confirm Password: •••••••••

Sign Up

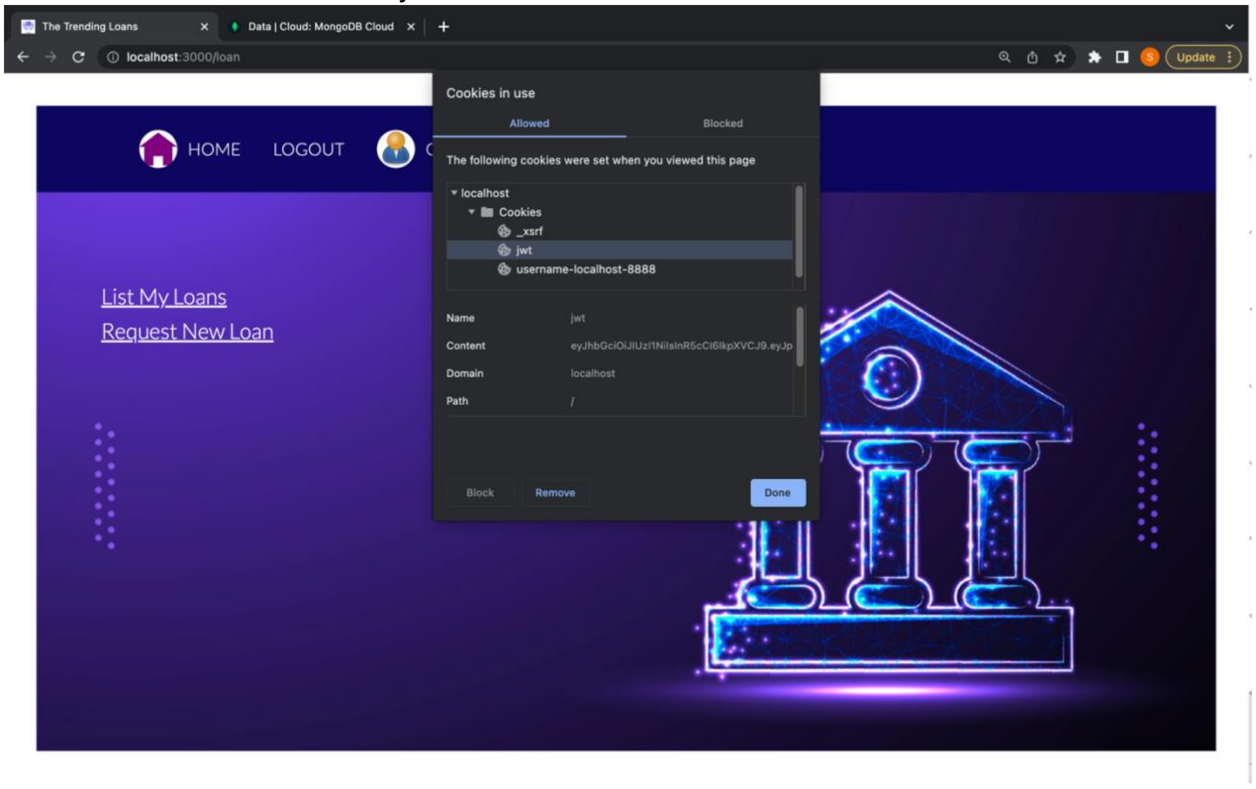
Login Current User:



After Logging In:



Authentication Cookies with a jwt has been set:

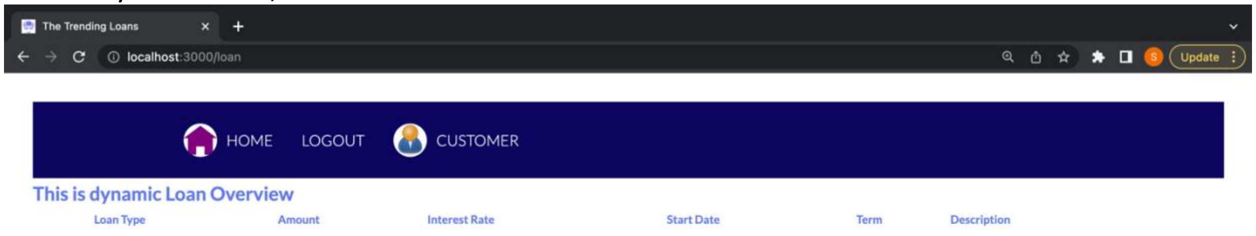


The following cookies were set when you viewed this page

localhost
Cookies
_xsrf
jwt
username-localhost-8888

Name	jwt
Content	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJp
Domain	localhost
Path	/

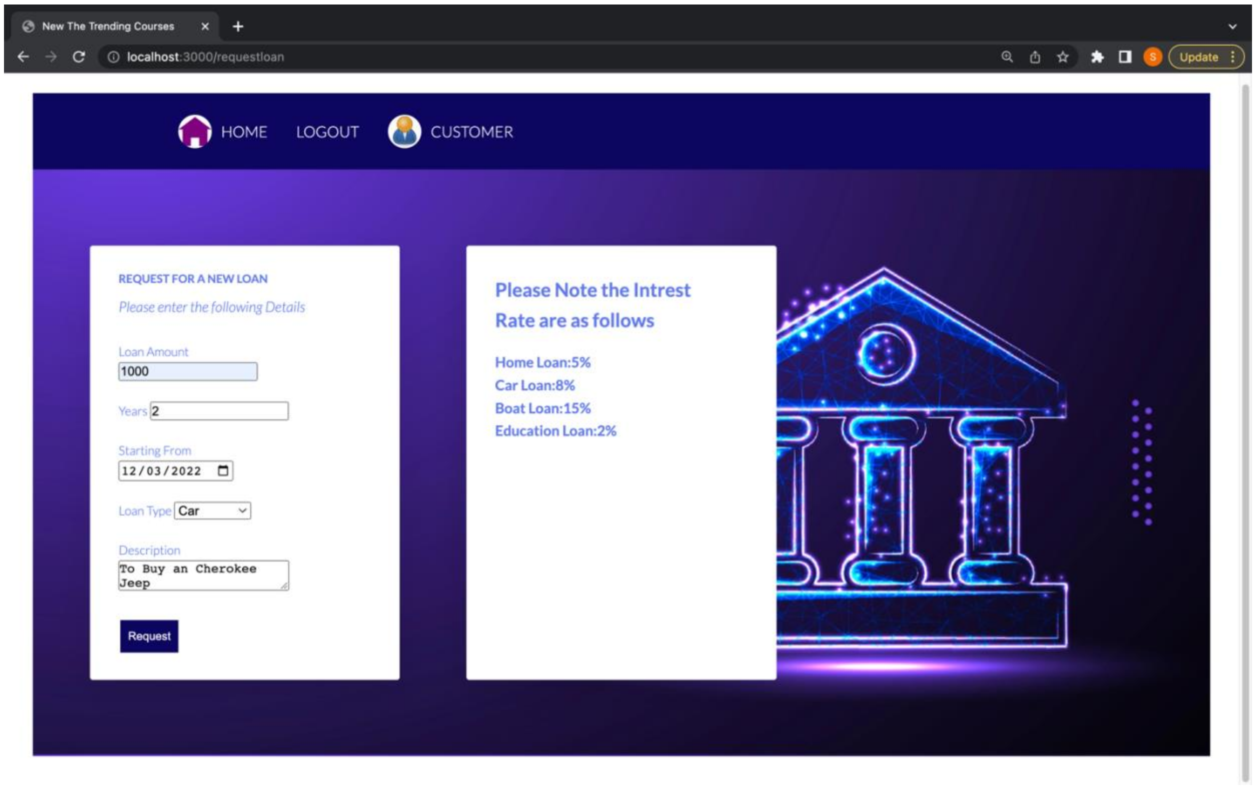
In List My Loans Link, there are no Loans for the new user:



This is dynamic Loan Overview

Loan Type	Amount	Interest Rate	Start Date	Term	Description

Create New Loans



REQUEST FOR A NEW LOAN

Please enter the following Details

Loan Amount 1000

Years 2

Starting From 12/03/2022

Loan Type Car

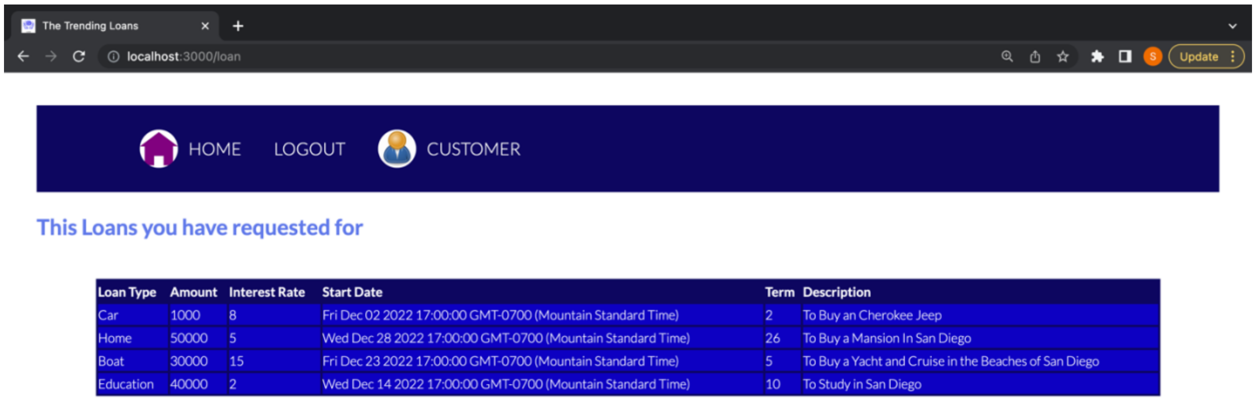
Description To Buy an Cherokee Jeep

Request

Please Note the Intrest Rate are as follows

Home Loan:5%
Car Loan:8%
Boat Loan:15%
Education Loan:2%

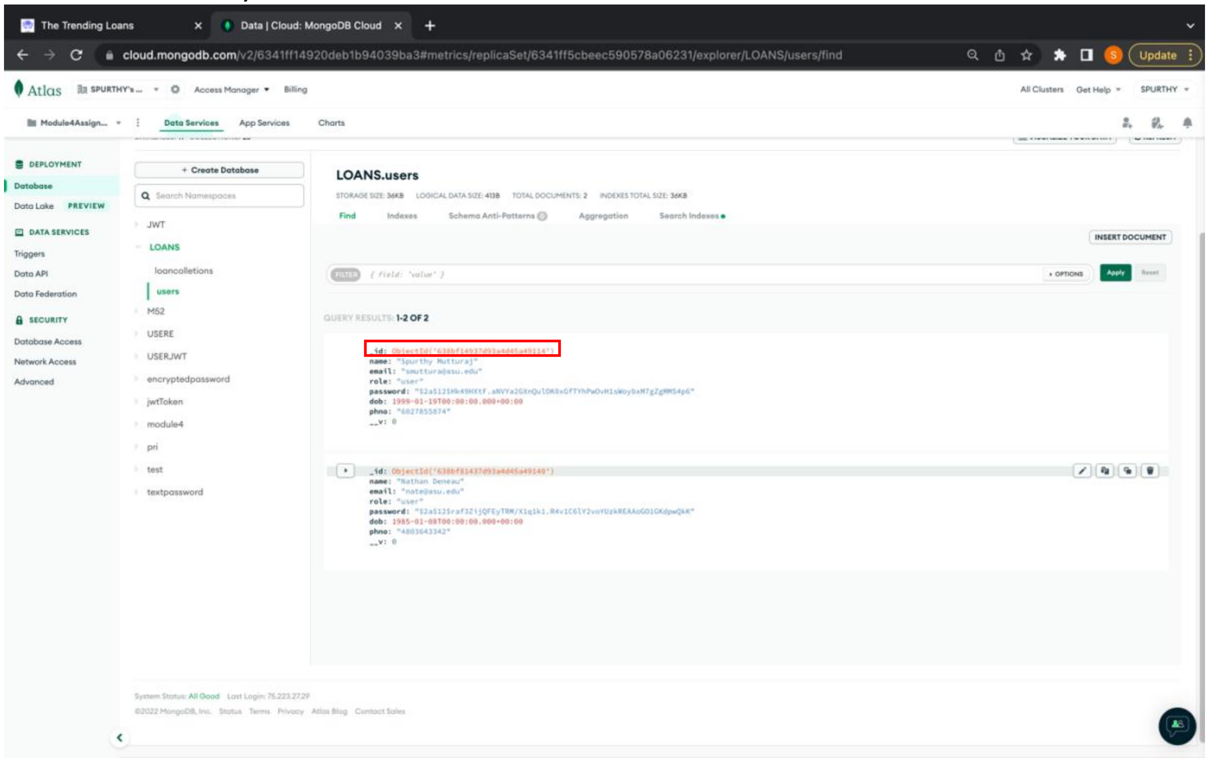
List Loan Details particular to a User



This Loans you have requested for

Loan Type	Amount	Interest Rate	Start Date	Term	Description
Car	1000	8	Fri Dec 02 2022 17:00:00 GMT-0700 (Mountain Standard Time)	2	To Buy an Cherokee Jeep
Home	50000	5	Wed Dec 28 2022 17:00:00 GMT-0700 (Mountain Standard Time)	26	To Buy a Mansion In San Diego
Boat	30000	15	Fri Dec 23 2022 17:00:00 GMT-0700 (Mountain Standard Time)	5	To Buy a Yacht and Cruise in the Beaches of San Diego
Education	40000	2	Wed Dec 14 2022 17:00:00 GMT-0700 (Mountain Standard Time)	10	To Study in San Diego

Verification:
There are currently 2 users in the Database:

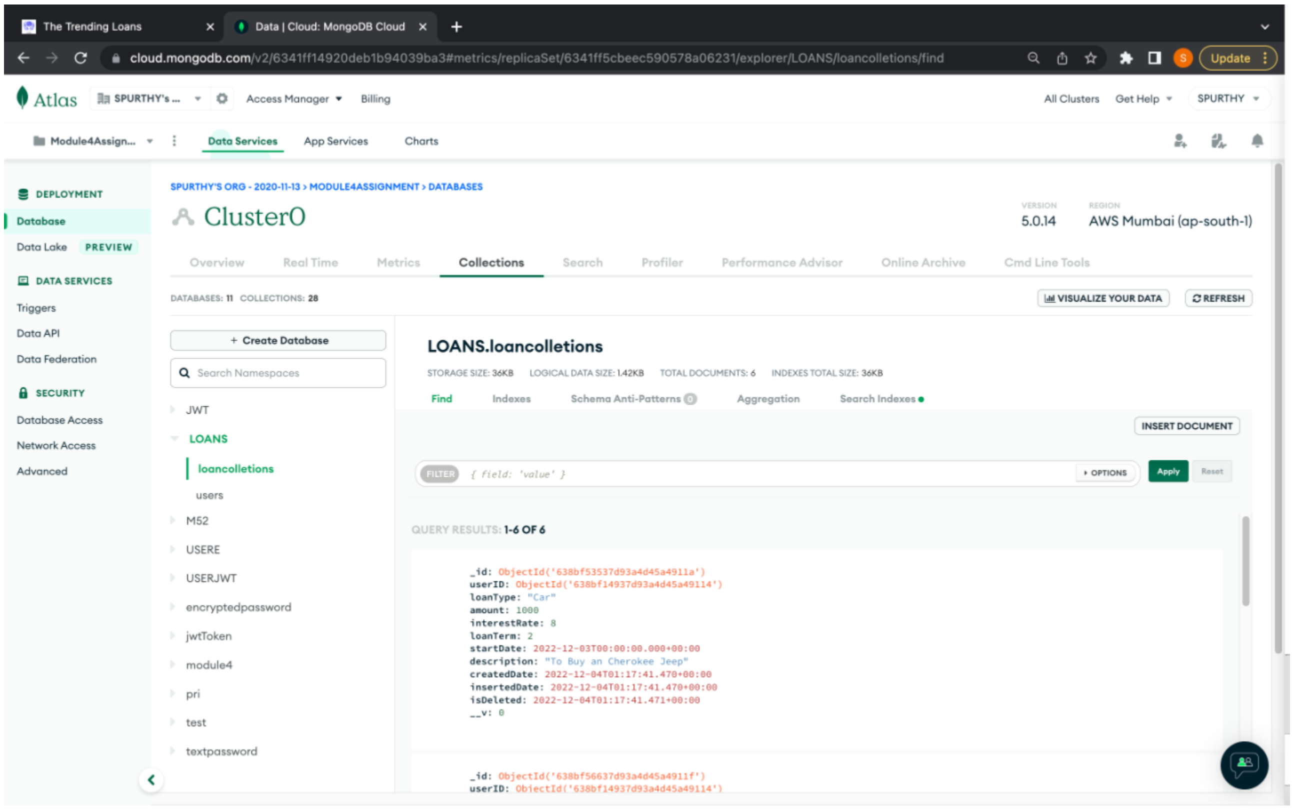


LOANS.users

QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('638bf14937d93a4d45a49114')
name: "Spurthy Mutturaj"
email: "smuttura@asu.edu"
role: "user"
password: "$2a$12$Hk49HXtF.aNVYa2GXnQul0K0xGfTYhPwOvHIsWoybxM7gZgMM54pG"
dob: 1999-01-19T00:00:00.000+00:00
phno: "6027855874"
__v: 0

_id: ObjectId('638bf81437d93a4d45a49140')
name: "Nathan Deneau"
email: "nate@asu.edu"
role: "user"
password: "$2a$12$raf3ZijQFEyTRM/X1q1k1.R4v1C6lY2voYUzkREAAoGO1GKdpwQkK"
dob: 1985-01-08T00:00:00.000+00:00
phno: "4803643342"
__v: 0
```

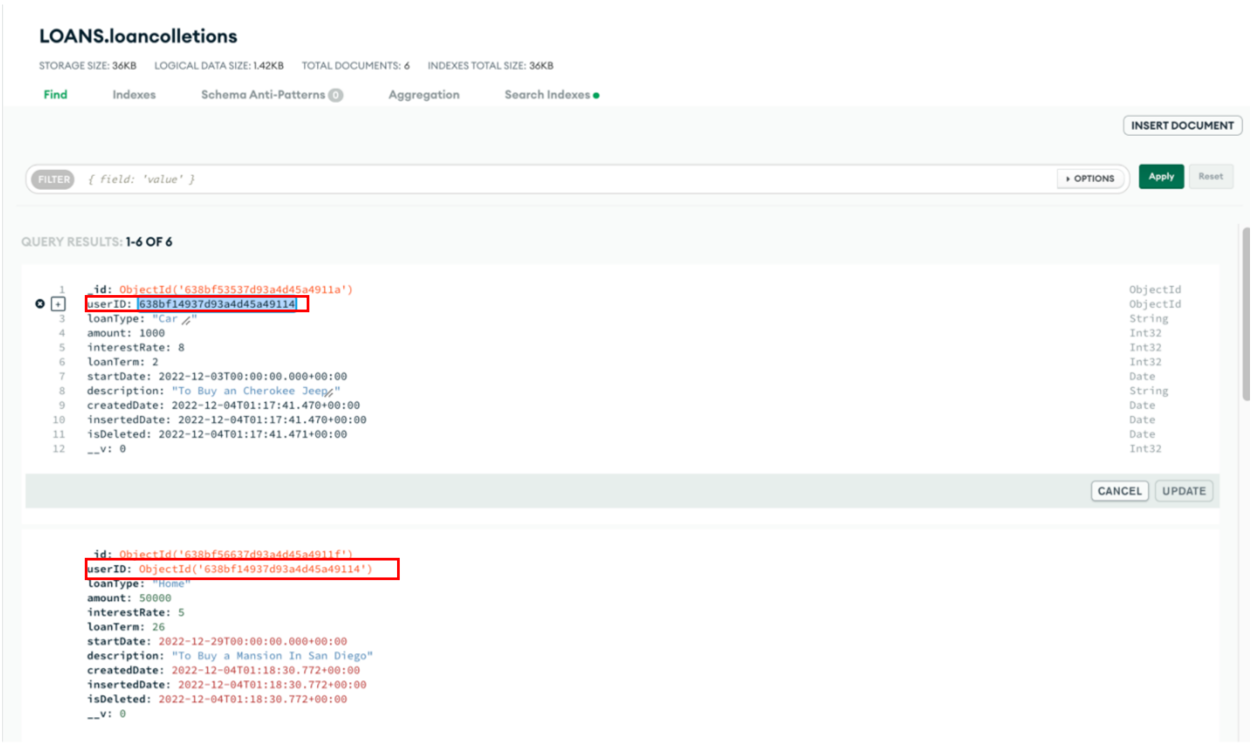
The Database collection contains 6 loans, but only 4 loans belong to the particular user with userID: 638bf14937d93a4d45a49114



LOANS.loancolletions

QUERY RESULTS: 1-6 OF 6

```
_id: ObjectId('638bf53537d93a4d45a4911a')
userID: ObjectId('638bf14937d93a4d45a49114')
loanType: "Car"
amount: 1000
interestRate: 8
loanTerm: 2
startDate: 2022-12-03T00:00:00.000+00:00
description: "To Buy an Cherokee Jeep"
createdDate: 2022-12-04T01:17:41.470+00:00
insertedDate: 2022-12-04T01:17:41.470+00:00
isDeleted: 2022-12-04T01:17:41.471+00:00
__v: 0

_id: ObjectId('638bf56637d93a4d45a4911f')
userID: ObjectId('638bf14937d93a4d45a49114')
```



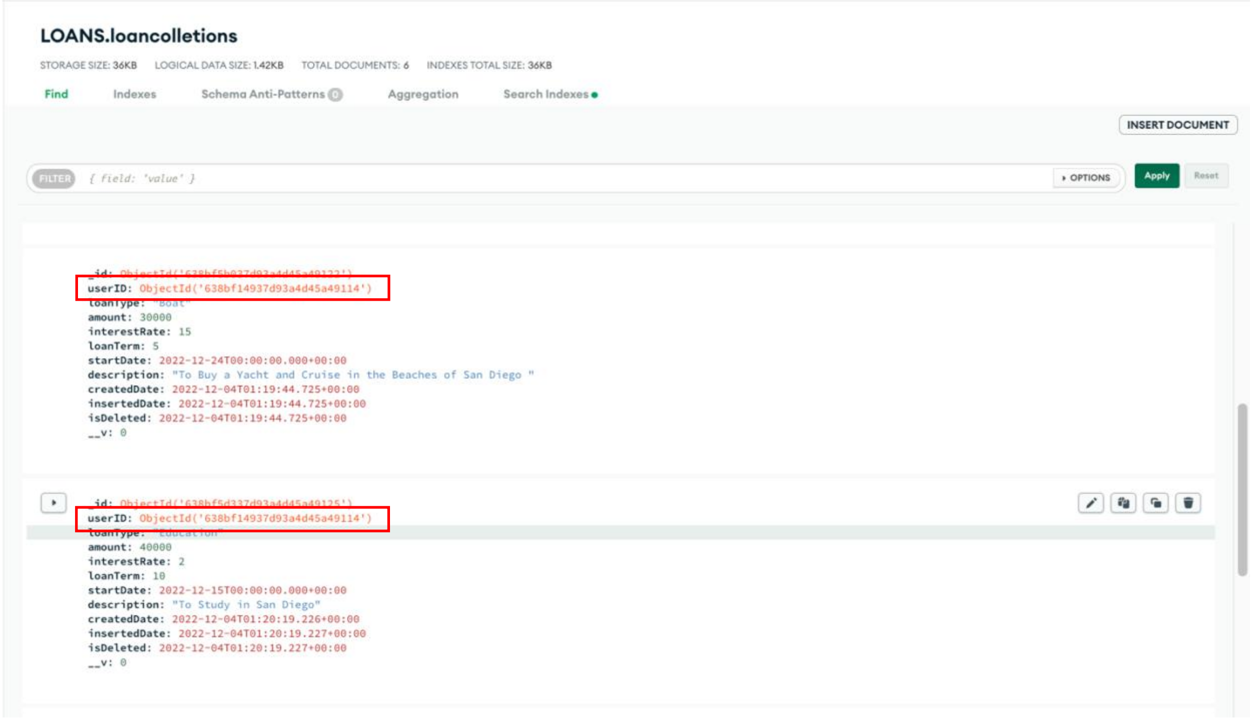
LOANS.loancolletions

STORAGE SIZE: 36KB   LOGICAL DATA SIZE: 1.42KB   TOTAL DOCUMENTS: 6   INDEXES TOTAL SIZE: 36KB

QUERY RESULTS: 1-6 OF 6

```
_id: ObjectId('638bf53537d93a4d45a4911a')                           ObjectId
userID: 638bf14937d93a4d45a49114                                    ObjectId
loanType: "Car"                                                     String
amount: 1000                                                        Int32
interestRate: 8                                                     Int32
loanTerm: 2                                                         Int32
startDate: 2022-12-03T00:00:00.000+00:00                            Date
description: "To Buy an Cherokee Jeep"                              String
createdDate: 2022-12-04T01:17:41.470+00:00                          Date
insertedDate: 2022-12-04T01:17:41.470+00:00                         Date
isDeleted: 2022-12-04T01:17:41.471+00:00                            Date
__v: 0                                                              Int32
```

```
_id: ObjectId('638bf56637d93a4d45a4911f')
userID: ObjectId('638bf14937d93a4d45a49114')
loanType: "Home"
amount: 50000
interestRate: 5
loanTerm: 26
startDate: 2022-12-29T00:00:00.000+00:00
description: "To Buy a Mansion In San Diego"
createdDate: 2022-12-04T01:18:30.772+00:00
insertedDate: 2022-12-04T01:18:30.772+00:00
isDeleted: 2022-12-04T01:18:30.772+00:00
__v: 0
```

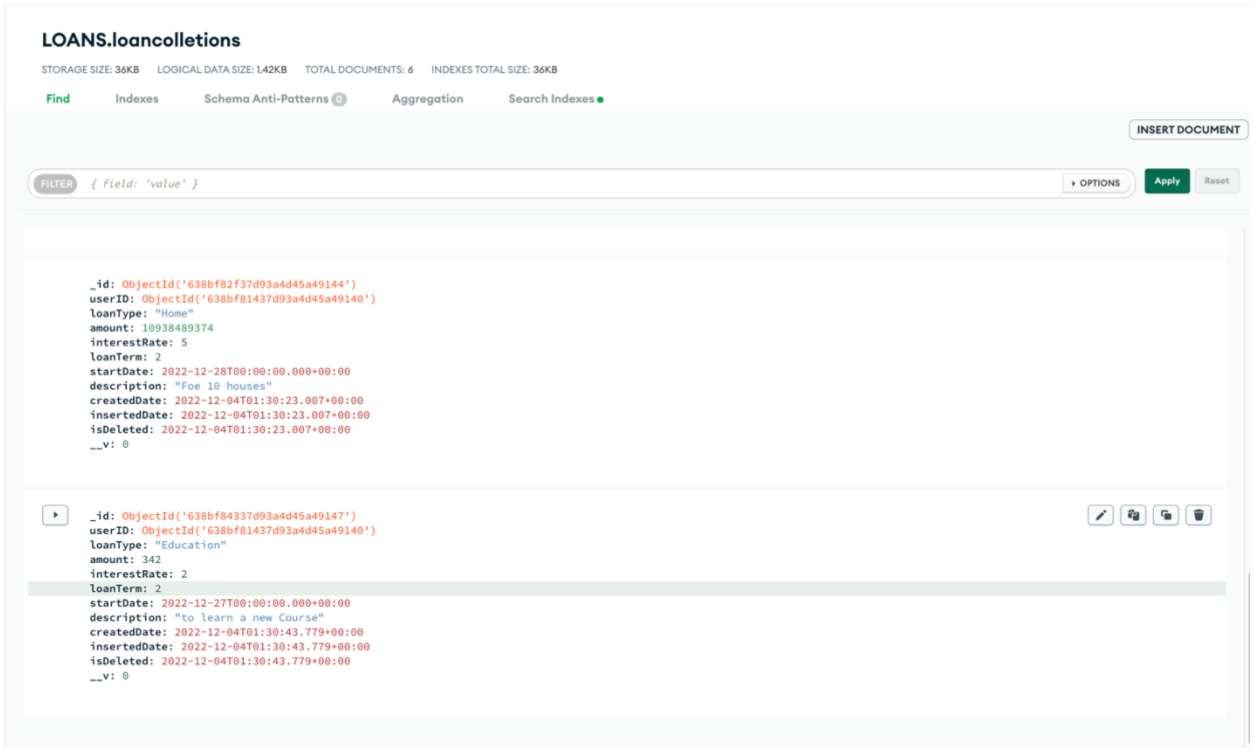


LOANS.loancolletions

STORAGE SIZE: 36KB   LOGICAL DATA SIZE: 1.42KB   TOTAL DOCUMENTS: 6   INDEXES TOTAL SIZE: 36KB

```
_id: ObjectId('638bf5b037d93a4d45a49122')
userID: ObjectId('638bf14937d93a4d45a49114')
loanType: "Boat"
amount: 30000
interestRate: 15
loanTerm: 5
startDate: 2022-12-24T00:00:00.000+00:00
description: "To Buy a Yacht and Cruise in the Beaches of San Diego "
createdDate: 2022-12-04T01:19:44.725+00:00
insertedDate: 2022-12-04T01:19:44.725+00:00
isDeleted: 2022-12-04T01:19:44.725+00:00
__v: 0
```

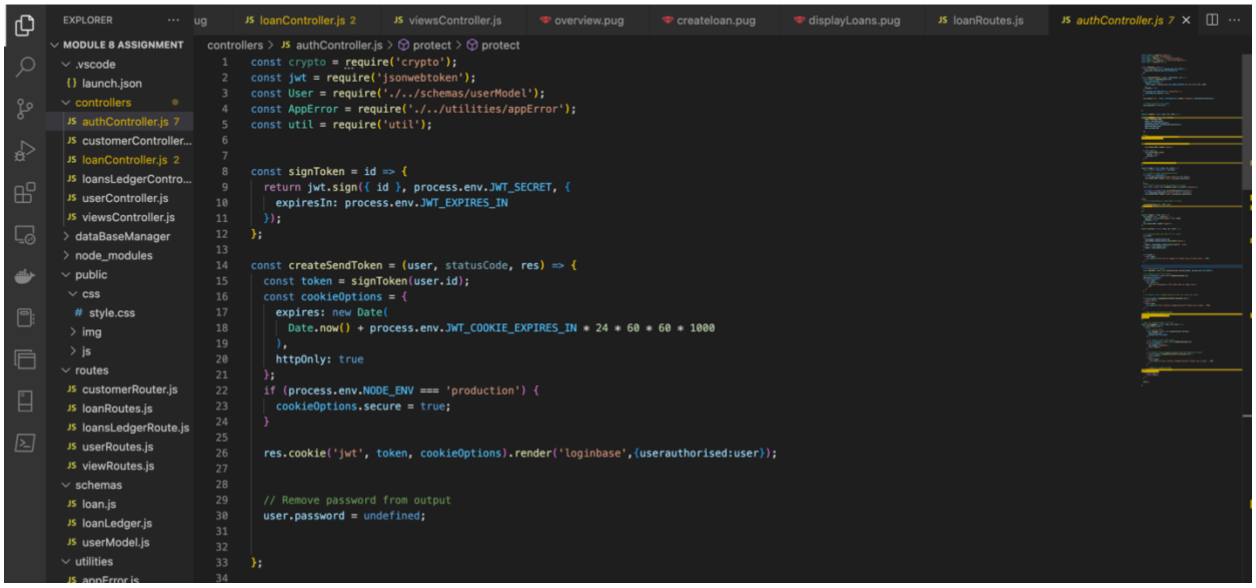
```
_id: ObjectId('638bf5d337d93a4d45a49125')
userID: ObjectId('638bf14937d93a4d45a49114')
loanType: "Education"
amount: 40000
interestRate: 2
loanTerm: 10
startDate: 2022-12-15T00:00:00.000+00:00
description: "To Study in San Diego"
createdDate: 2022-12-04T01:20:19.226+00:00
insertedDate: 2022-12-04T01:20:19.227+00:00
isDeleted: 2022-12-04T01:20:19.227+00:00
__v: 0
```



```
LOANS.loancolletions
STORAGE SIZE: 36KB   LOGICAL DATA SIZE: 1.42KB   TOTAL DOCUMENTS: 6   INDEXES TOTAL SIZE: 36KB

_id: ObjectId('638bf82f37d93a4d45a49144')
userID: ObjectId('638bf81437d93a4d45a49140')
loanType: "Home"
amount: 10938489374
interestRate: 5
loanTerm: 2
startDate: 2022-12-28T00:00:00.000+00:00
description: "Foe 10 houses"
createdDate: 2022-12-04T01:30:23.007+00:00
insertedDate: 2022-12-04T01:30:23.007+00:00
isDeleted: 2022-12-04T01:30:23.007+00:00
__v: 0

_id: ObjectId('638bf84337d93a4d45a49147')
userID: ObjectId('638bf81437d93a4d45a49140')
loanType: "Education"
amount: 342
interestRate: 2
loanTerm: 2
startDate: 2022-12-27T00:00:00.000+00:00
description: "to learn a new Course"
createdDate: 2022-12-04T01:30:43.779+00:00
insertedDate: 2022-12-04T01:30:43.779+00:00
isDeleted: 2022-12-04T01:30:43.779+00:00
__v: 0
```

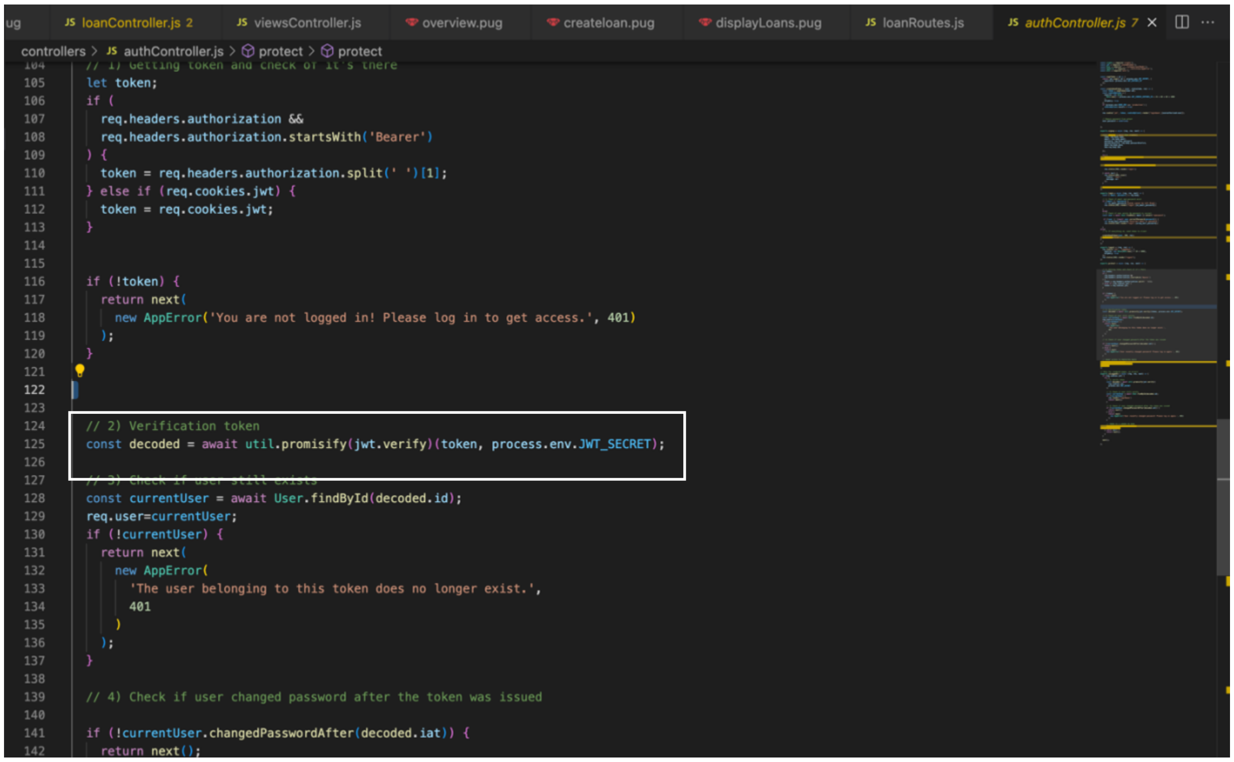
B. MiddleWare

Authentication(creating a token)



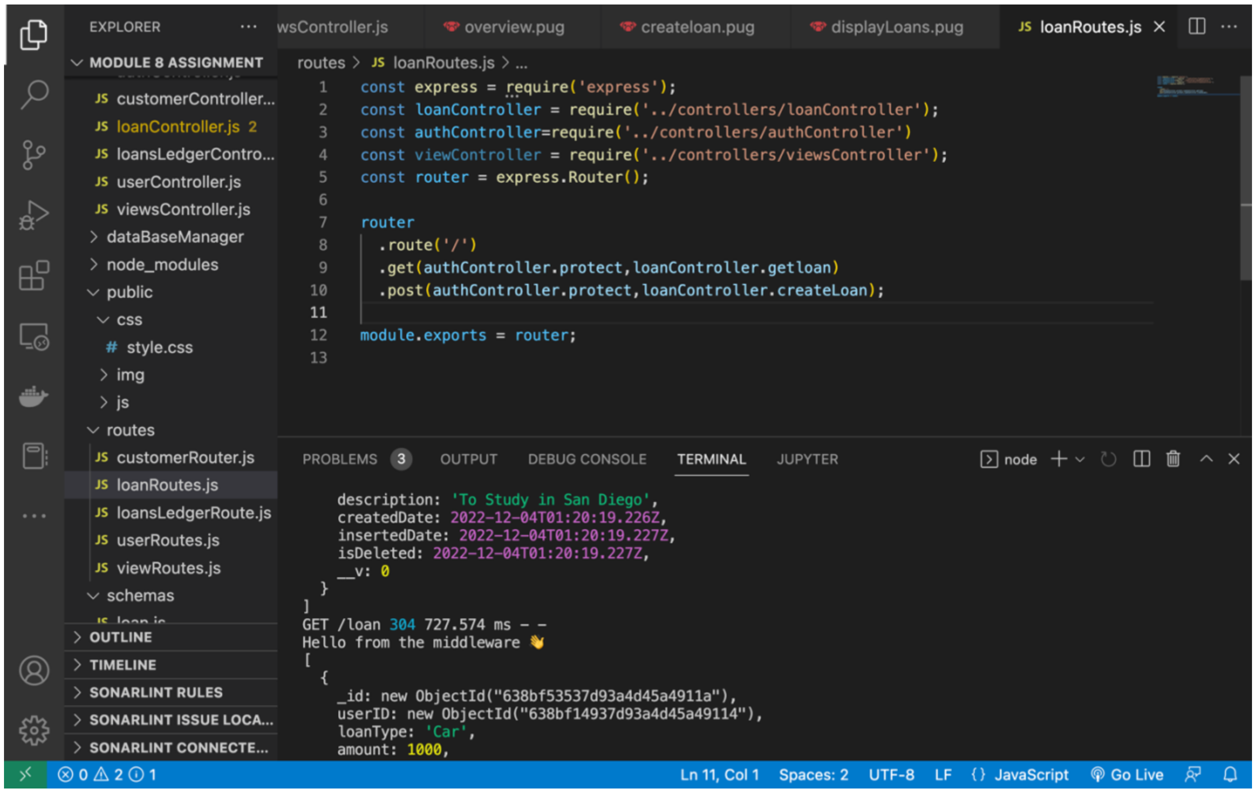
```
controllers > authController.js > protect > protect
const crypto = require('crypto');
const jwt = require('jsonwebtoken');
const User = require('./../schemas/userModel');
const AppError = require('./../utilities/appError');
const util = require('util');


const signToken = id => {
  return jwt.sign({ id }, process.env.JWT_SECRET, {
    expiresIn: process.env.JWT_EXPIRES_IN
  });
};

const createSendToken = (user, statusCode, res) => {
  const token = signToken(user.id);
  const cookieOptions = {
    expires: new Date(
      Date.now() + process.env.JWT_COOKIE_EXPIRES_IN * 24 * 60 * 60 * 1000
    ),
    httpOnly: true
  };
  if (process.env.NODE_ENV === 'production') {
    cookieOptions.secure = true;
  }

  res.cookie('jwt', token, cookieOptions).render('loginbase',{userauthorised:user});


  // Remove password from output
  user.password = undefined;


};
```

Authorisation(using jwt.verify)



```javascript
  // 1) Getting token and check of it's there
  let token;
  if (
    req.headers.authorization &&
    req.headers.authorization.startsWith('Bearer')
  ) {
    token = req.headers.authorization.split(' ')[1];
  } else if (req.cookies.jwt) {
    token = req.cookies.jwt;
  }


  if (!token) {
    return next(
      new AppError('You are not logged in! Please log in to get access.', 401)
    );
  }



  // 2) Verification token
  const decoded = await util.promisify(jwt.verify)(token, process.env.JWT_SECRET);

  // 3) Check if user still exists
  const currentUser = await User.findById(decoded.id);
  req.user=currentUser;
  if (!currentUser) {
    return next(
      new AppError(
        'The user belonging to this token does no longer exist.',
        401
      )
    );
  }

  // 4) Check if user changed password after the token was issued

  if (!currentUser.changedPasswordAfter(decoded.iat)) {
    return next();
```

Protected Routes

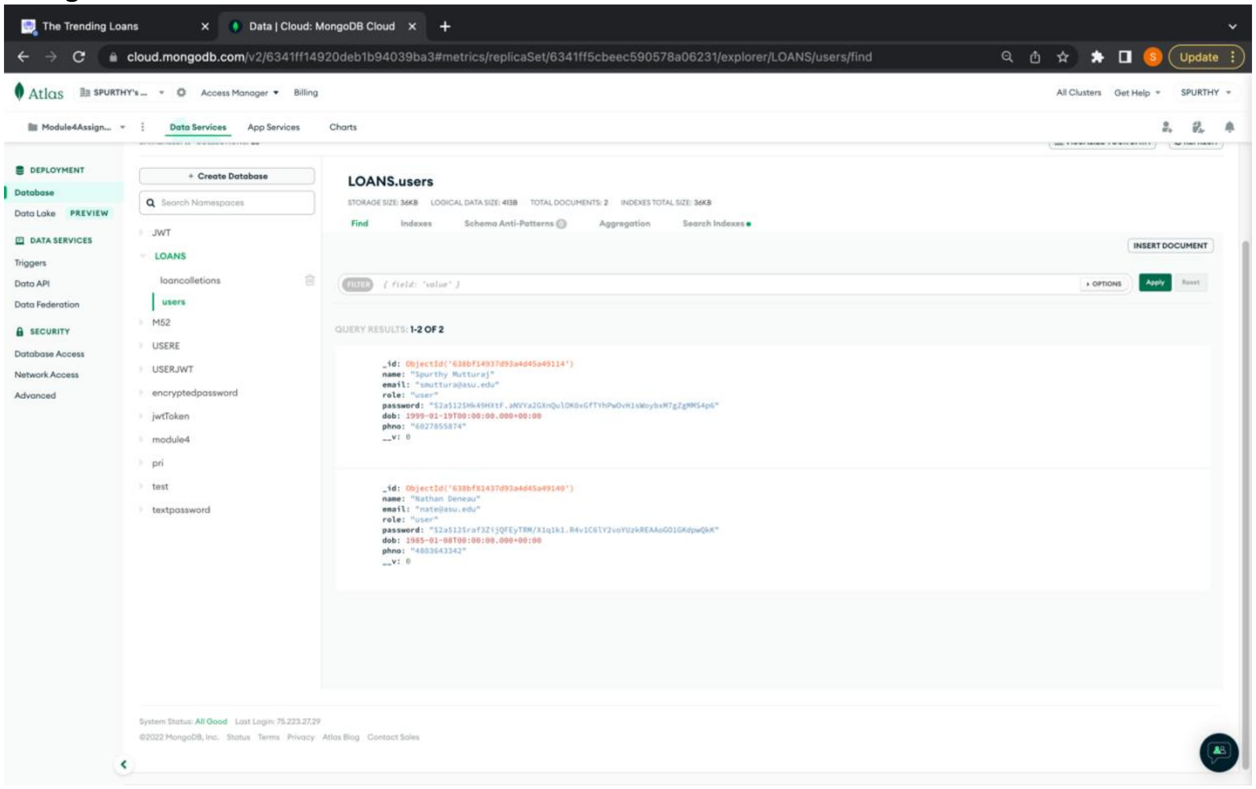


```javascript
const express = require('express');
const loanController = require('../controllers/loanController');
const authController=require('../controllers/authController')
const viewController = require('../controllers/viewsController');
const router = express.Router();

router
  .route('/')
  .get(authController.protect,loanController.getloan)
  .post(authController.protect,loanController.createLoan);

module.exports = router;
```

```
    description: 'To Study in San Diego',
    createdDate: 2022-12-04T01:20:19.226Z,
    insertedDate: 2022-12-04T01:20:19.227Z,
    isDeleted: 2022-12-04T01:20:19.227Z,
    __v: 0
  }
]
GET /loan 304 727.574 ms - -
Hello from the middleware 👋
[
  {
    _id: new ObjectId("638bf53537d93a4d45a4911a"),
    userID: new ObjectId("638bf14937d93a4d45a49114"),
    loanType: 'Car',
    amount: 1000,
```

Backend:

MongoDB Database LOANS



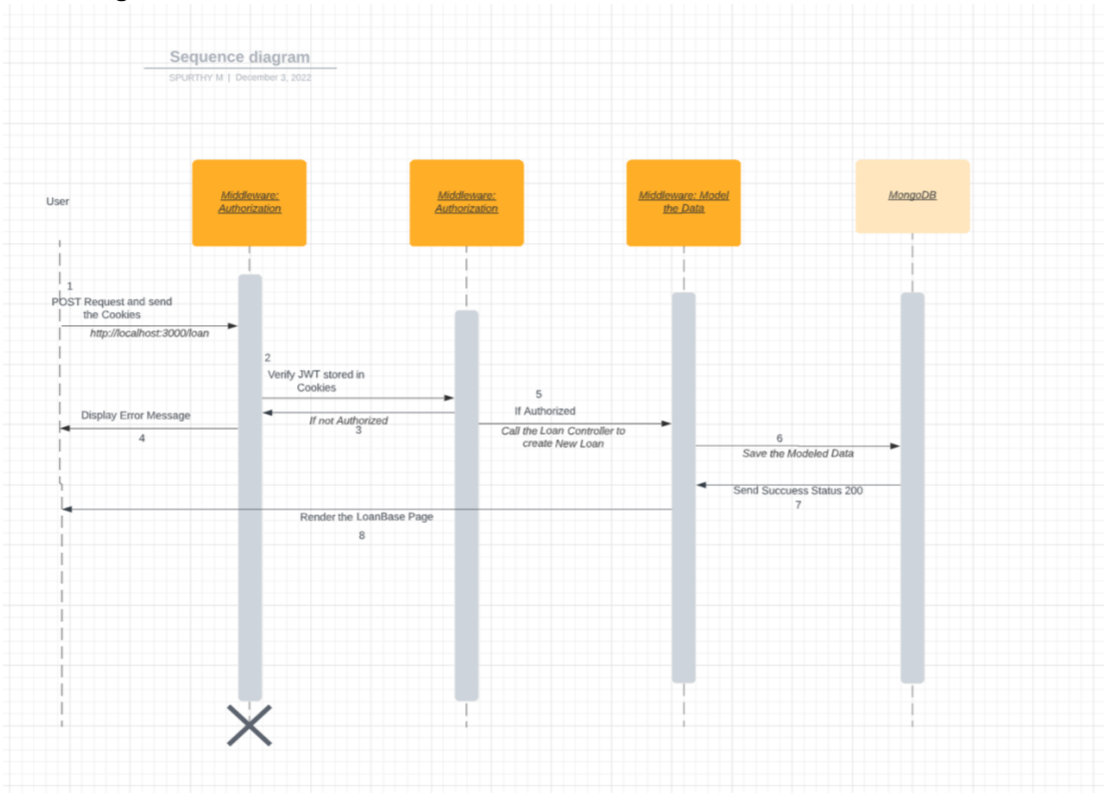
LOANS.users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 413B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 36KB

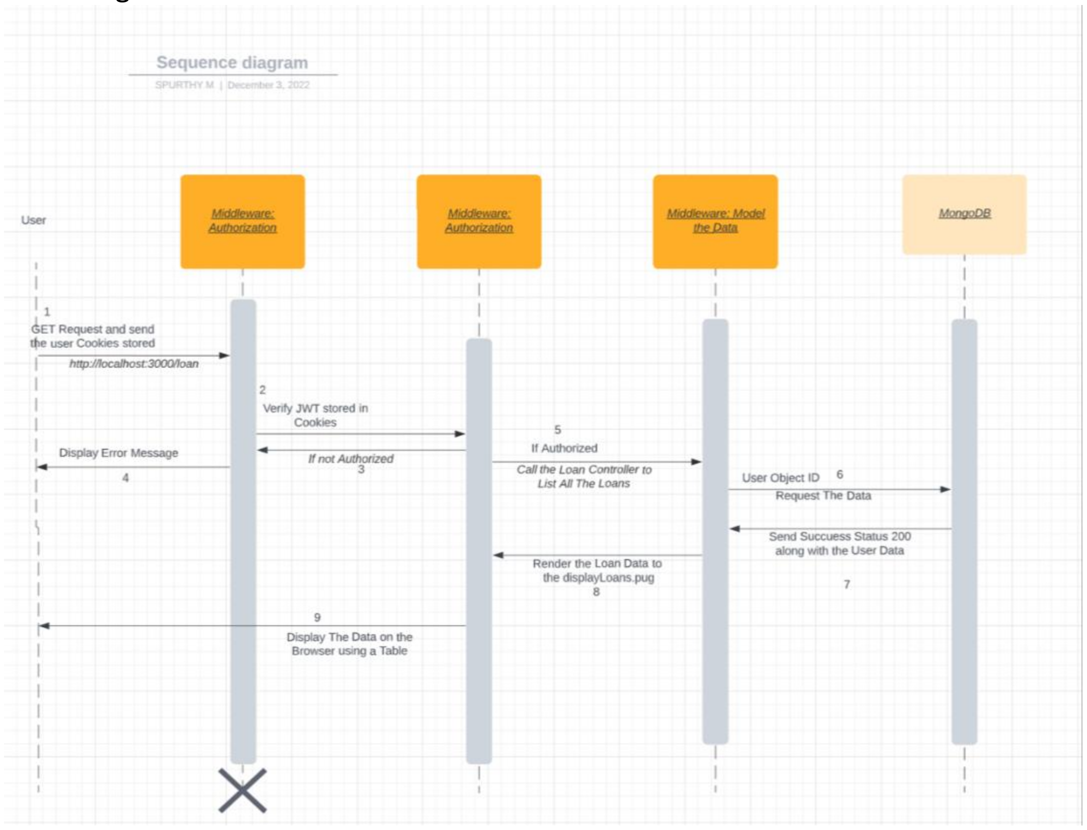
QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('638bf14937d93a4d45a49114')
name: "Spurthy Mutturaj"
email: "smuttura@asu.edu"
role: "user"
password: "$2a$12$Hk49HXtF.aNVYa2GXnQulOK0xGfTYhPwOvH1sWoybxM7gZgMM54p6"
dob: 1999-01-19T00:00:00.000+00:00
phno: "6027855874"
__v: 0

_id: ObjectId('638bf81437d93a4d45a49140')
name: "Nathan Deneau"
email: "nate@asu.edu"
role: "user"
password: "$2a$12$raf3ZijQFEyTRM/X1q1k1.R4v1C6lY2voYUzkREAAoGO1GKdpwQkK"
dob: 1985-01-08T00:00:00.000+00:00
phno: "4803643342"
__v: 0
```

Flow Diagram for Create Loan :

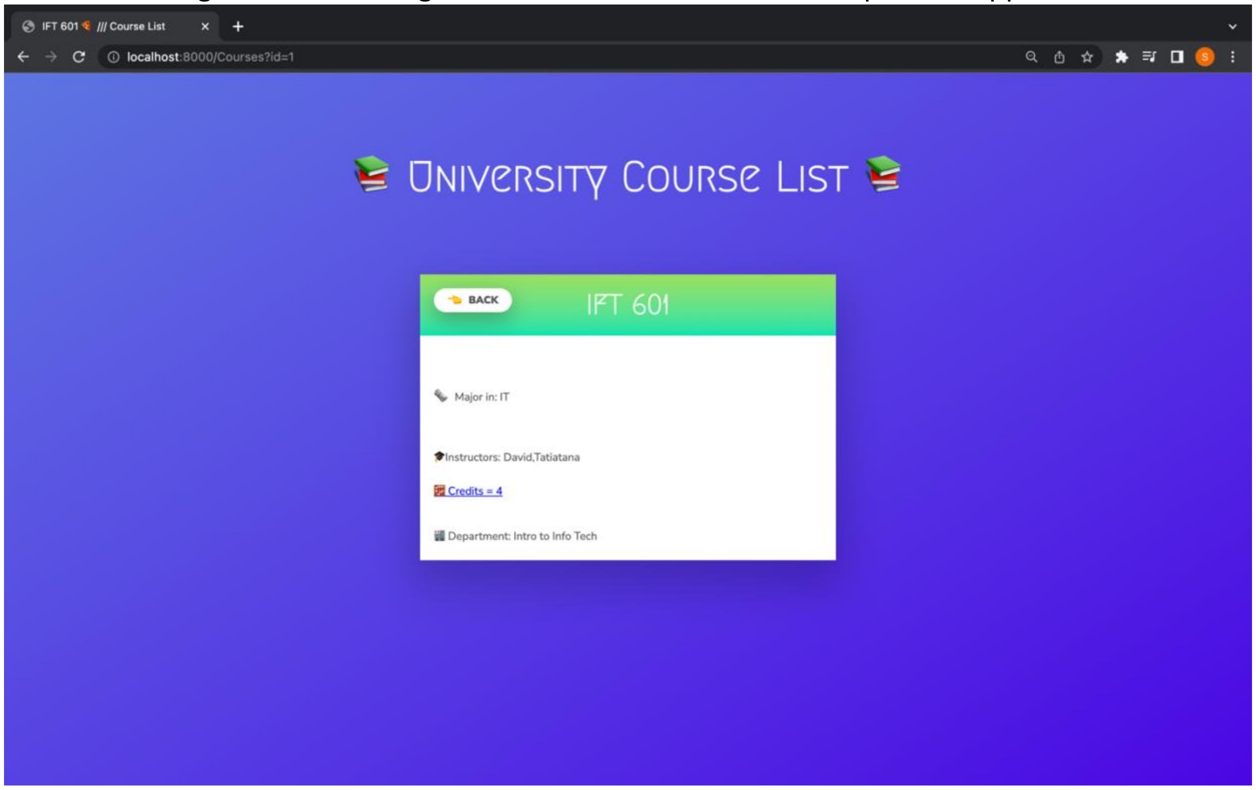

Sequence diagram
SPURTHY M | December 3, 2022
User
Middleware: Authorization
Middleware: Authorization
Middleware: Model the Data
MongoDB
1
POST Request and send the Cookies
http://localhost:3000/loan
2
Verify JWT stored in Cookies
If not Authorized
3
Display Error Message
4
5
If Authorized
Call the Loan Controller to create New Loan
6
Save the Modeled Data
Send Succuess Status 200
7
Render the LoanBase Page
8

Flow Diagram for List All Loans of a User


Sequence diagram
SPURTHY M. | December 3, 2022
User
Middleware: Authorization
Middleware: Authorization
Middleware: Model the Data
MongoDB
1
GET Request and send the user Cookies stored
http://localhost:3000/loan
2
Verify JWT stored in Cookies
If not Authorized
3
Display Error Message
4
5
If Authorized
Call the Loan Controller to List All The Loans
User Object ID
6
Request The Data
Send Succuess Status 200 along with the User Data
7
Render the Loan Data to the displayLoans.pug
8
9
Display The Data on the Browser using a Table

Progress From Module1 through Module 8:

The Course began with installing the Visual Studio Code and a simple web application



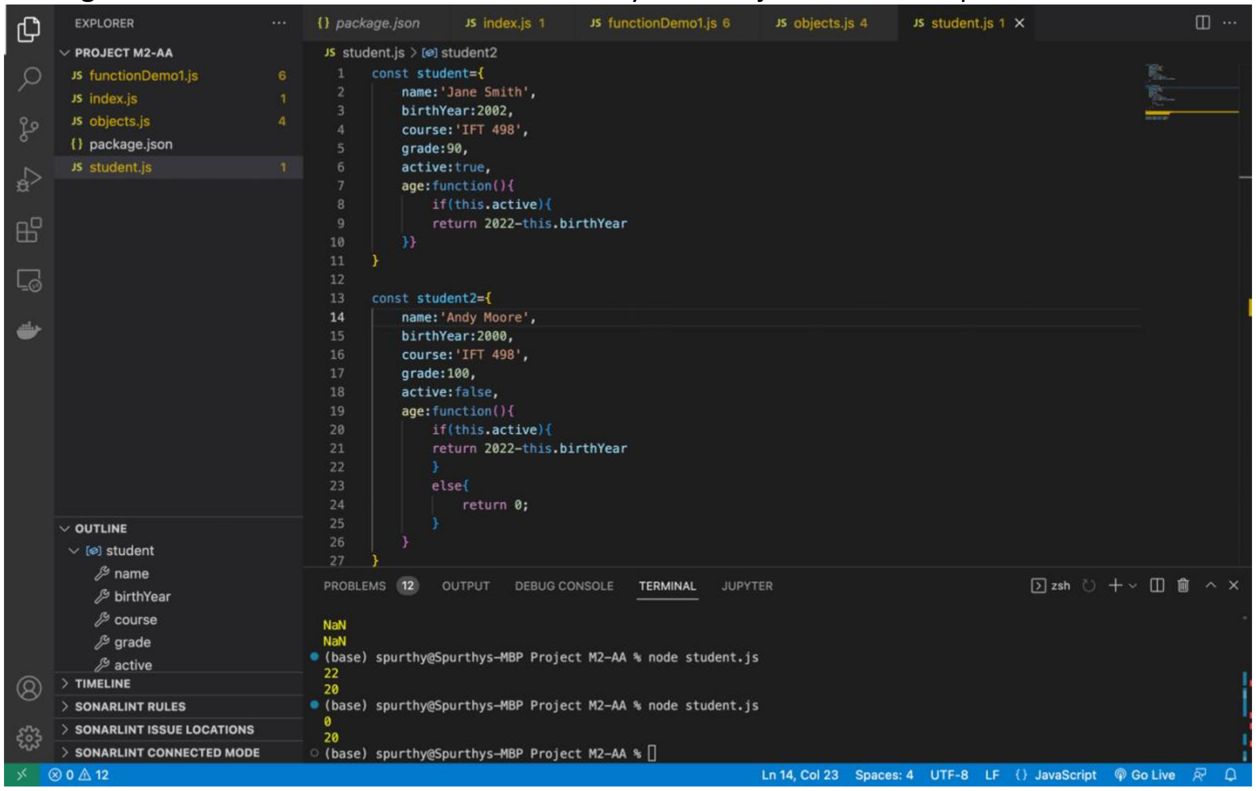
Major in: IT

Instructors: David,Tatiatana

Credits = 4

Department: Intro to Info Tech

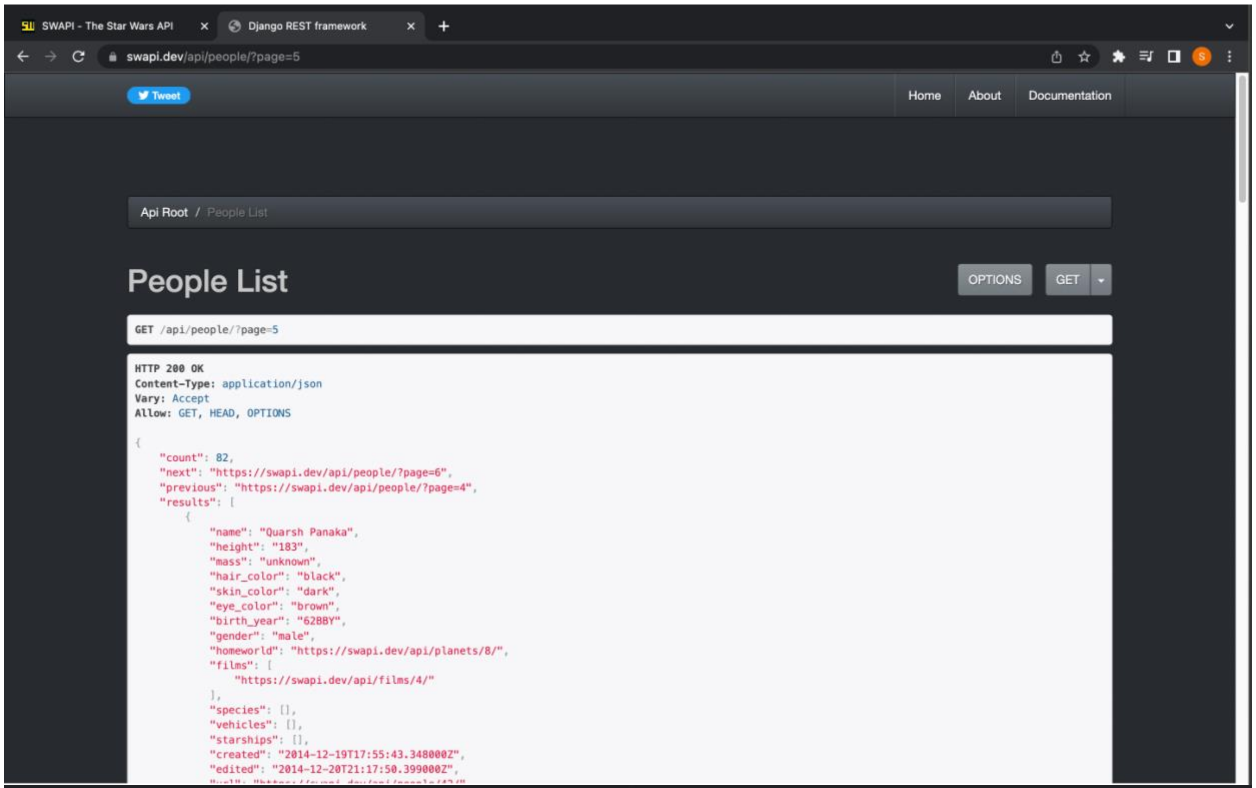
Through Module 2 we learnt about ES6 and arrays and Objects in JavaScript



```
const student={
    name:'Jane Smith',
    birthYear:2002,
    course:'IFT 498',
    grade:90,
    active:true,
    age:function(){
        if(this.active){
        return 2022-this.birthYear
    }}
}

const student2={
    name:'Andy Moore',
    birthYear:2000,
    course:'IFT 498',
    grade:100,
    active:false,
    age:function(){
        if(this.active){
        return 2022-this.birthYear
        }
        else{
            return 0;
        }
    }
}
```

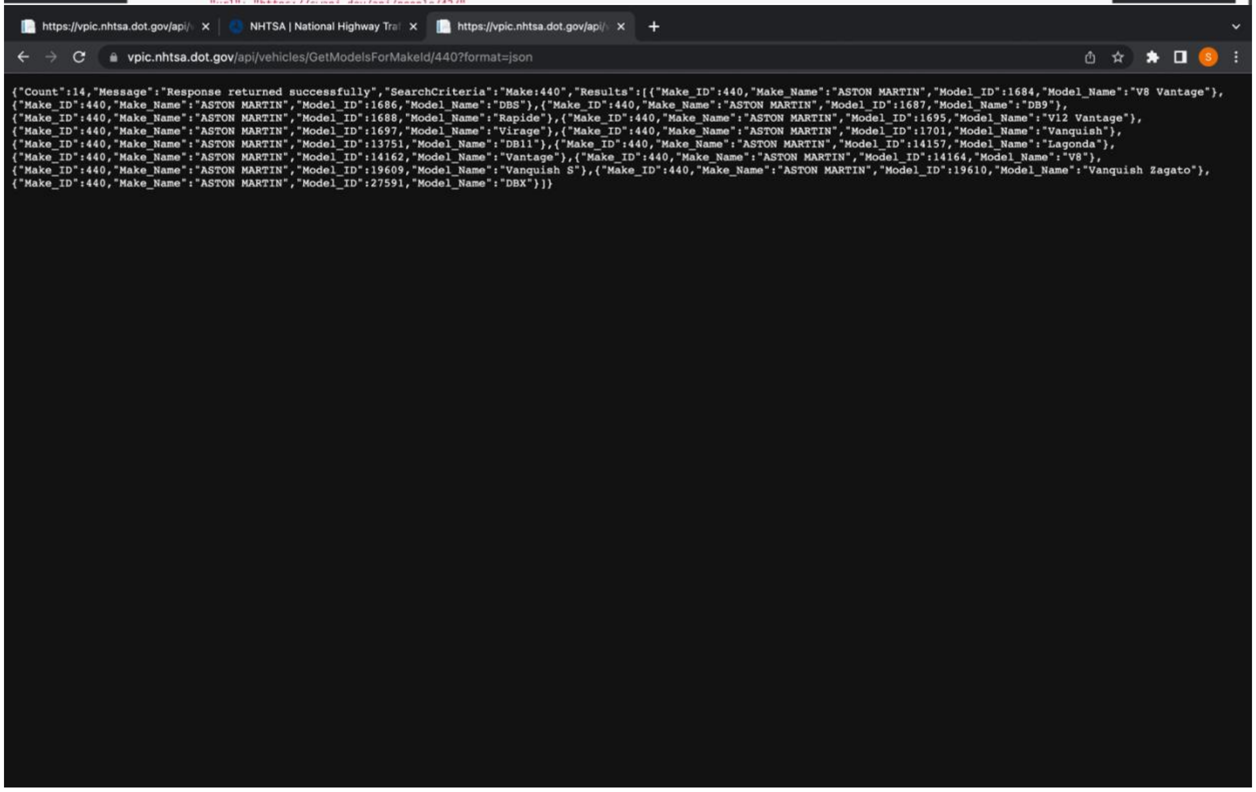
```
NaN
NaN
(base) spurthy@Spurthys-MBP Project M2-AA % node student.js
22
20
(base) spurthy@Spurthys-MBP Project M2-AA % node student.js
0
20
(base) spurthy@Spurthys-MBP Project M2-AA %
```

In Module3, we explored swapi.dev api and data formats like XML and JSON



```
GET /api/people/?page=5
```

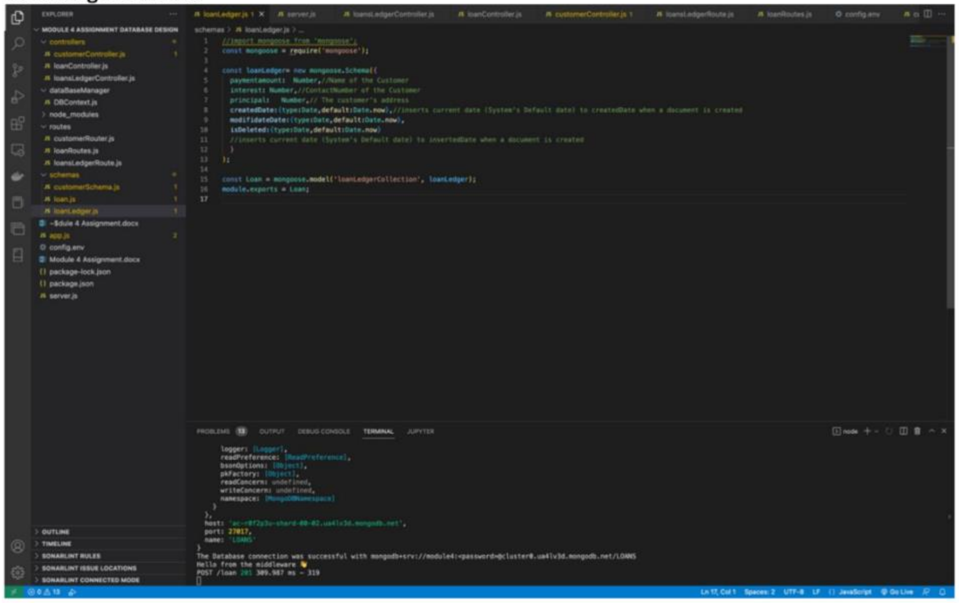
```
HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, HEAD, OPTIONS

{
    "count": 82,
    "next": "https://swapi.dev/api/people/?page=6",
    "previous": "https://swapi.dev/api/people/?page=4",
    "results": [
        {
            "name": "Quarsh Panaka",
            "height": "183",
            "mass": "unknown",
            "hair_color": "black",
            "skin_color": "dark",
            "eye_color": "brown",
            "birth_year": "62BBY",
            "gender": "male",
            "homeworld": "https://swapi.dev/api/planets/8/",
            "films": [
                "https://swapi.dev/api/films/4/"
            ],
            "species": [],
            "vehicles": [],
            "starships": [],
            "created": "2014-12-19T17:55:43.348000Z",
            "edited": "2014-12-20T21:17:50.399000Z",
```


https://vpic.nhtsa.dot.gov/api/
NHTSA | National Highway Tra
https://vpic.nhtsa.dot.gov/api/
vpic.nhtsa.dot.gov/api/vehicles/GetModelsForMakeId/440?format=json

```
{"Count":14,"Message":"Response returned successfully","SearchCriteria":"Make:440","Results":[{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":1684,"Model_Name":"V8 Vantage"},
{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":1686,"Model_Name":"DBS"},{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":1687,"Model_Name":"DB9"},
{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":1688,"Model_Name":"Rapide"},{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":1695,"Model_Name":"V12 Vantage"},
{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":1697,"Model_Name":"Virage"},{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":1701,"Model_Name":"Vanquish"},
{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":13751,"Model_Name":"DB11"},{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":14157,"Model_Name":"Lagonda"},
{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":14162,"Model_Name":"Vantage"},{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":14164,"Model_Name":"V8"},
{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":19609,"Model_Name":"Vanquish S"},{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":19610,"Model_Name":"Vanquish Zagato"},
{"Make_ID":440,"Make_Name":"ASTON MARTIN","Model_ID":27591,"Model_Name":"DBX"}]}
```

In Module 4, We learnt to connect to the MongoDb and access .

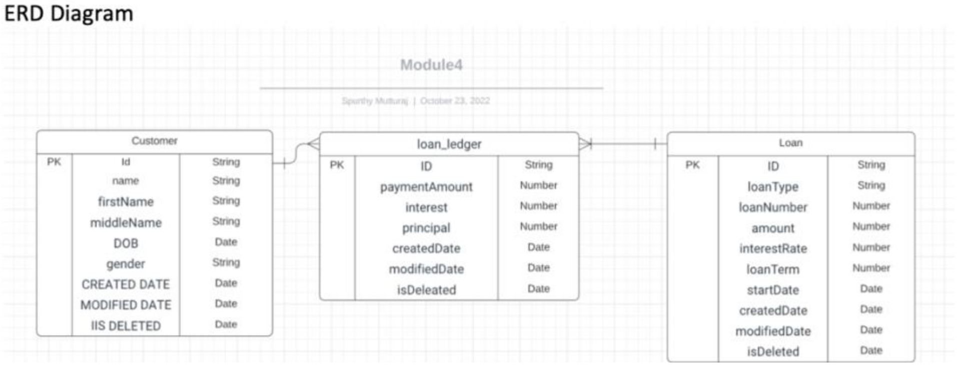
LoanLedger Schema



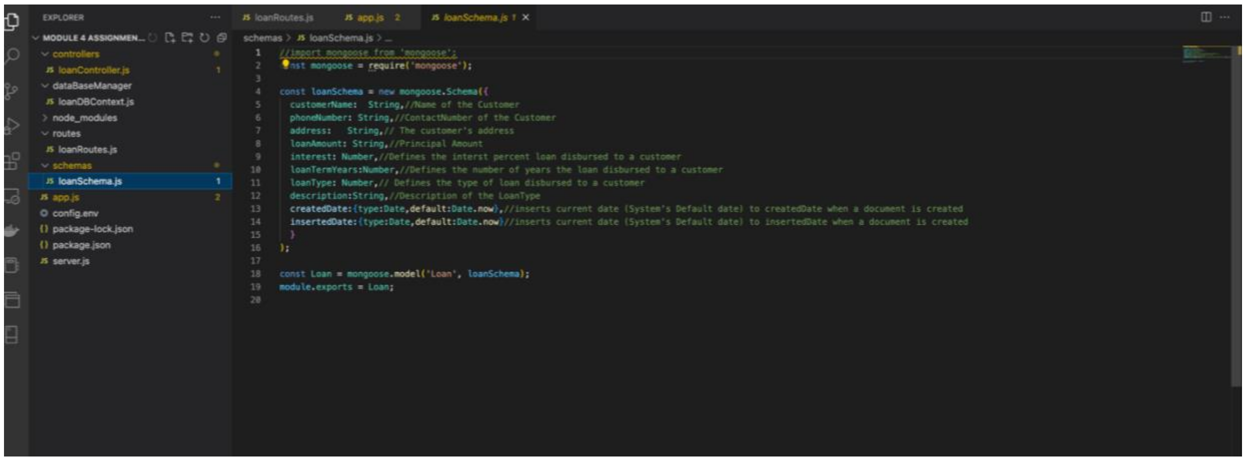
```
//import mongoose from 'mongoose';
const mongoose = require('mongoose');

const loanLedger= new mongoose.Schema({
  paymentamount:  Number,//Name of the Customer
  interest: Number,//ContactNumber of the Customer
  principal:   Number,// The customer's address
  createdDate:{type:Date,default:Date.now},//inserts current date (System's Default date) to createdDate when a document is created
  modifidateDate:{type:Date,default:Date.now},
  isDeleted:{type:Date,default:Date.now}
  //inserts current date (System's Default date) to insertedDate when a document is created
  }
);

const Loan = mongoose.model('loanLedgerCollection', loanLedger);
module.exports = Loan;
```

```
      logger: [Logger],
      readPreference: [ReadPreference],
      bsonOptions: [Object],
      pkFactory: [Object],
      readConcern: undefined,
      writeConcern: undefined,
      namespace: [MongoDBNamespace]
    }
  },
  host: 'ac-r0f2p3u-shard-00-02.ua4lv3d.mongodb.net',
  port: 27017,
  name: 'LOANS'
}
The Database connection was successful with mongodb+srv://module4:<password>@cluster0.ua4lv3d.mongodb.net/LOANS
Hello from the middleware 👋
POST /loan 201 309.987 ms - 319
```

ERD Diagram

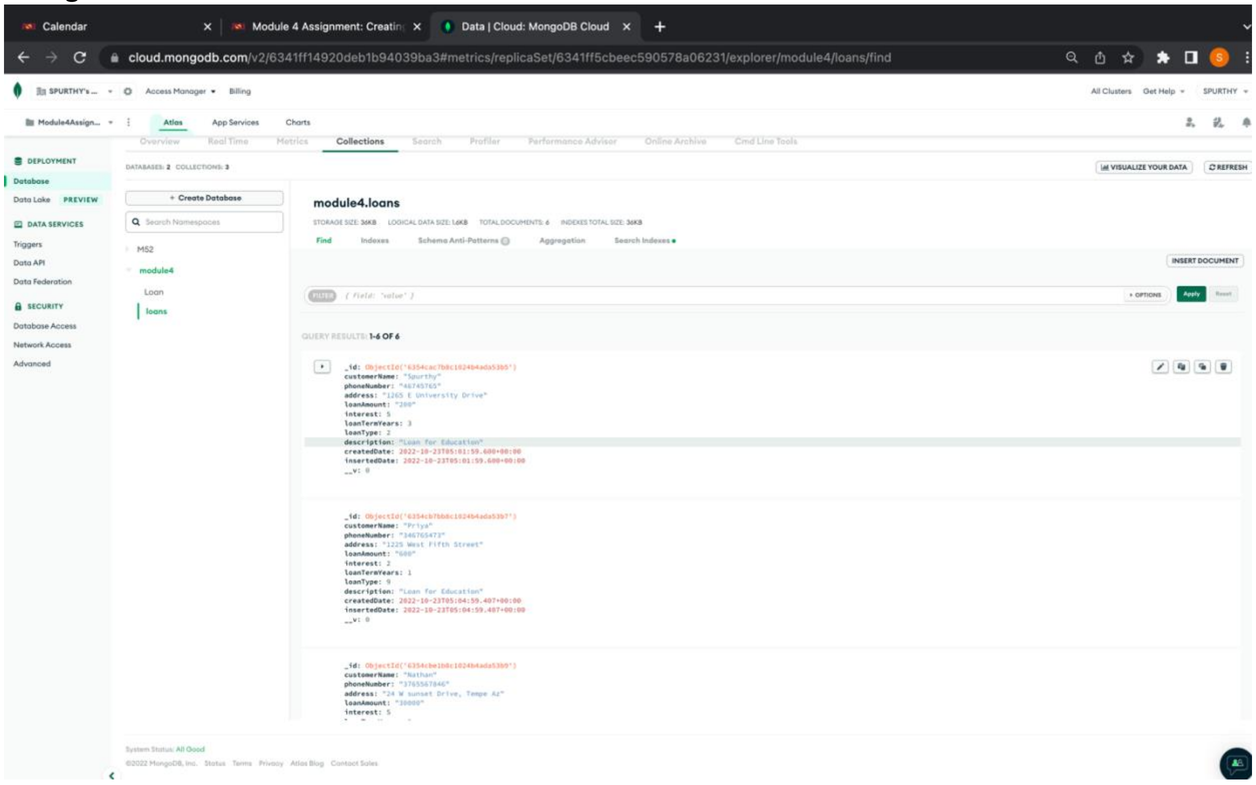


Mongoose Model to store Data:



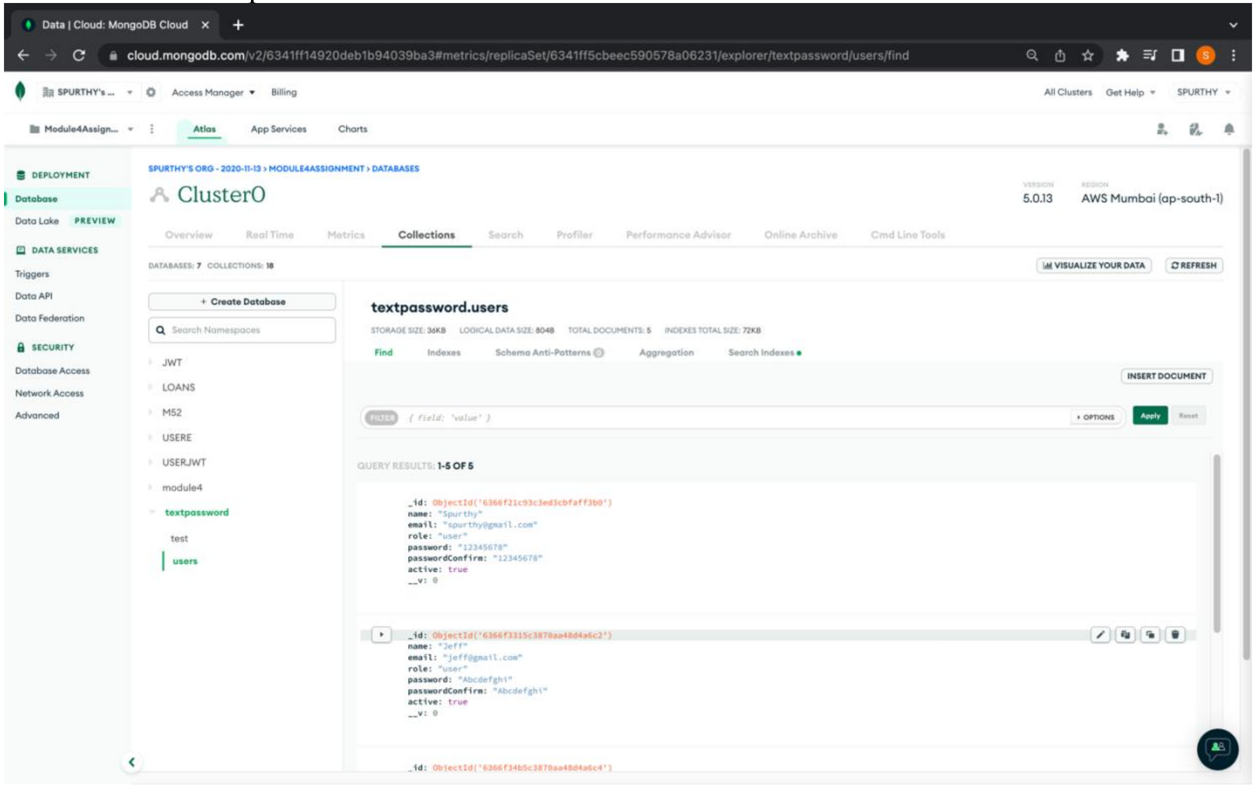
```
//import mongoose from 'mongoose';
const mongoose = require('mongoose');

const loanSchema = new mongoose.Schema({
  customerName:  String,//Name of the Customer
  phoneNumber: String,//ContactNumber of the Customer
  address:   String,// The customer's address
  loanAmount: String,//Principal Amount
  interest: Number,//Defines the interst percent loan disbursed to a customer
  loanTermYears:Number,//Defines the number of years the loan disbursed to a customer
  loanType: Number,// Defines the type of loan disbursed to a customer
  description:String,//Description of the LoanType
  createdDate:{type:Date,default:Date.now},//inserts current date (System's Default date) to createdDate when a document is created
  insertedDate:{type:Date,default:Date.now}//inserts current date (System's Default date) to insertedDate when a document is created
  }
);

const Loan = mongoose.model('Loan', loanSchema);
module.exports = Loan;
```

MongoDB:



```
_id: ObjectId('6354cac7b8c1024b4ada53b5')
customerName: "Spurthy"
phoneNumber: "46745765"
address: "1265 E University Drive"
loanAmount: "200"
interest: 5
loanTermYears: 3
loanType: 2
description: "Loan for Education"
createdDate: 2022-10-23T05:01:59.600+00:00
insertedDate: 2022-10-23T05:01:59.600+00:00
__v: 0

_id: ObjectId('6354cb7bb8c1024b4ada53b7')
customerName: "Priya"
phoneNumber: "346765473"
address: "1225 West Fifth Street"
loanAmount: "600"
interest: 2
loanTermYears: 1
loanType: 9
description: "Loan for Education"
createdDate: 2022-10-23T05:04:59.407+00:00
insertedDate: 2022-10-23T05:04:59.407+00:00
__v: 0

_id: ObjectId('6354cbe1b8c1024b4ada53b9')
customerName: "Nathan"
phoneNumber: "3765567846"
address: "24 W sunset Drive, Tempe Az"
loanAmount: "30000"
interest: 5
```

In Module 5 we learnt to store Data and sensitive information in 3 ways: as plain text, Encrypt it and also authenticate a User using JASON Web Token
5 users in DB "textpassword"



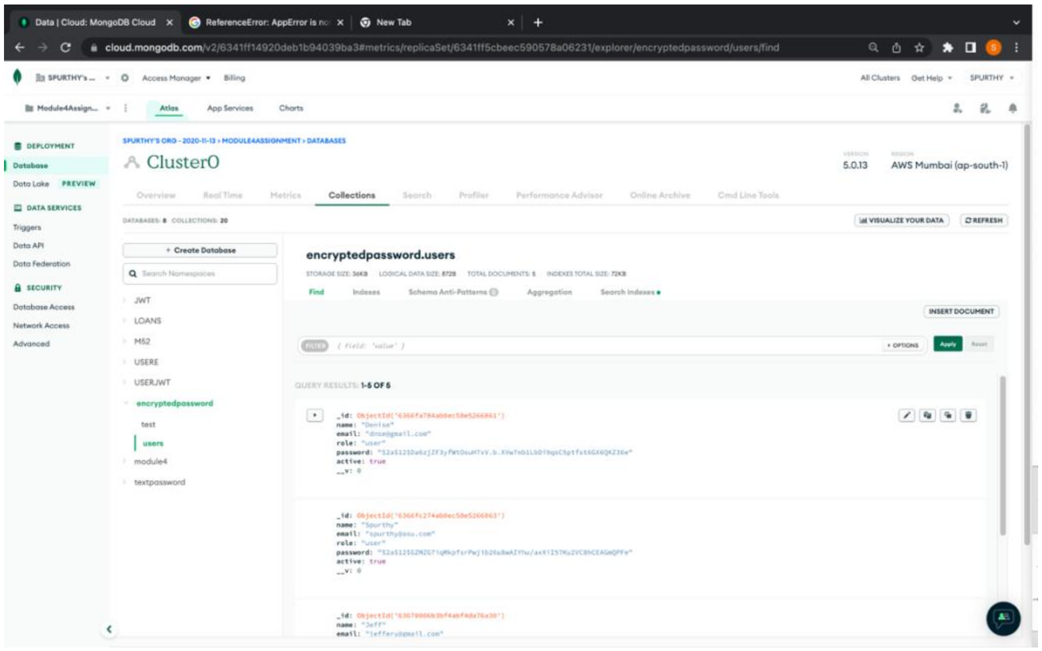
textpassword.users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 804B TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 72KB

QUERY RESULTS: 1-5 OF 5

```
_id: ObjectId('6366f21c93c3ed3cbfaff3b0')
name: "Spurthy"
email: "spurthy@gmail.com"
role: "user"
password: "12345678"
passwordConfirm: "12345678"
active: true
__v: 0

_id: ObjectId('6366f3315c3870aa48d4a6c2')
name: "Jeff"
email: "jeff@gmail.com"
role: "user"
password: "Abcdefghi"
passwordConfirm: "Abcdefghi"
active: true
__v: 0

_id: ObjectId('6366f34b5c3870aa48d4a6c4')
```

Database –"encryptedpassword" Screenshot



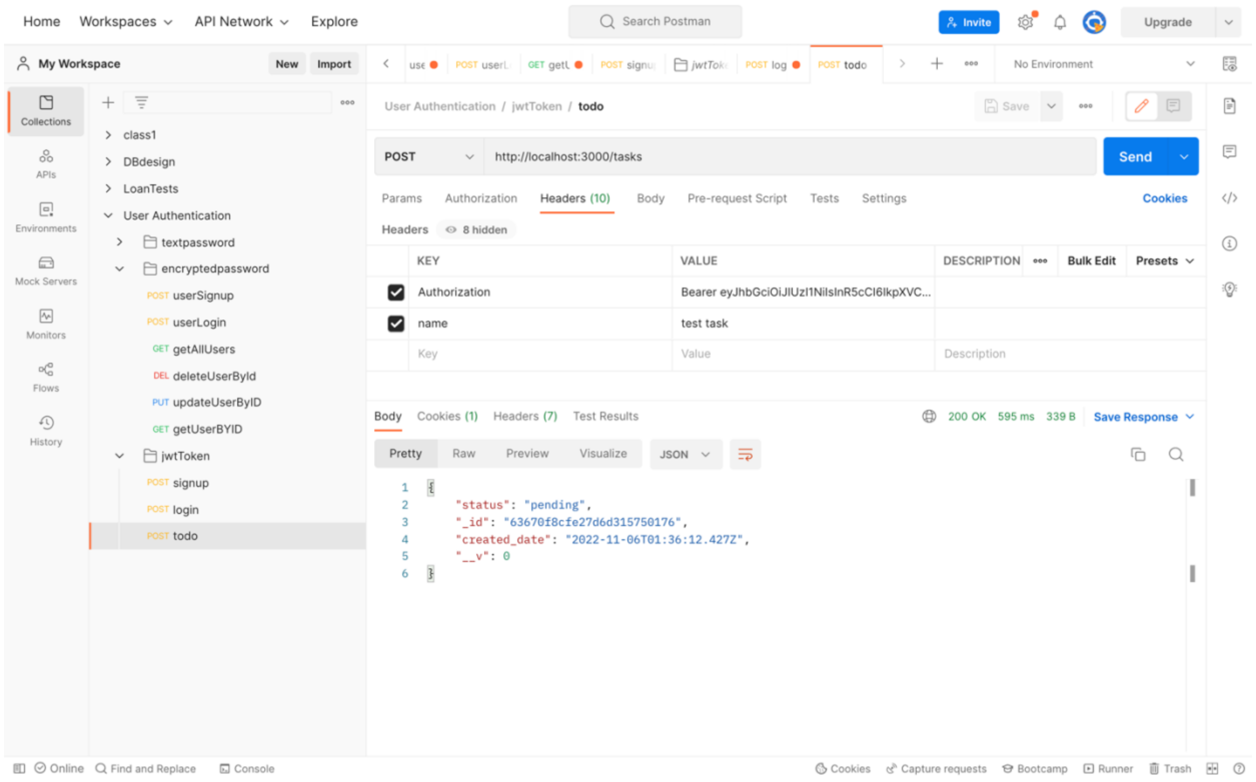
encryptedpassword.users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 872B TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 72KB

QUERY RESULTS: 1-5 OF 5

```
_id: ObjectId('6366fa784ab0ec58e5266861')
name: "Denise"
email: "dnse@gmail.com"
role: "user"
password: "$2a$12$Da6zjZF3yfWtOcuHTxV.b.XVw7nb1LbDl9qsC5ptfst6GX6QKZ36e"
active: true
__v: 0

_id: ObjectId('6366fc274ab0ec58e5266863')
name: "Spurthy"
email: "spurthy@asu.com"
role: "user"
password: "$2a$12$GZMZG7iqMkpfsrPwj1b26u8wAIYhu/axXiI5TKu2VC8hCEAGmQPFe"
active: true
__v: 0

_id: ObjectId('636790060b3bf4abf4da76a38')
name: "Jeff"
email: "jeffery@gmail.com"
```

Verifying the JWT created is functional by adding a task :



POST http://localhost:3000/tasks

User Authentication / jwtToken / todo

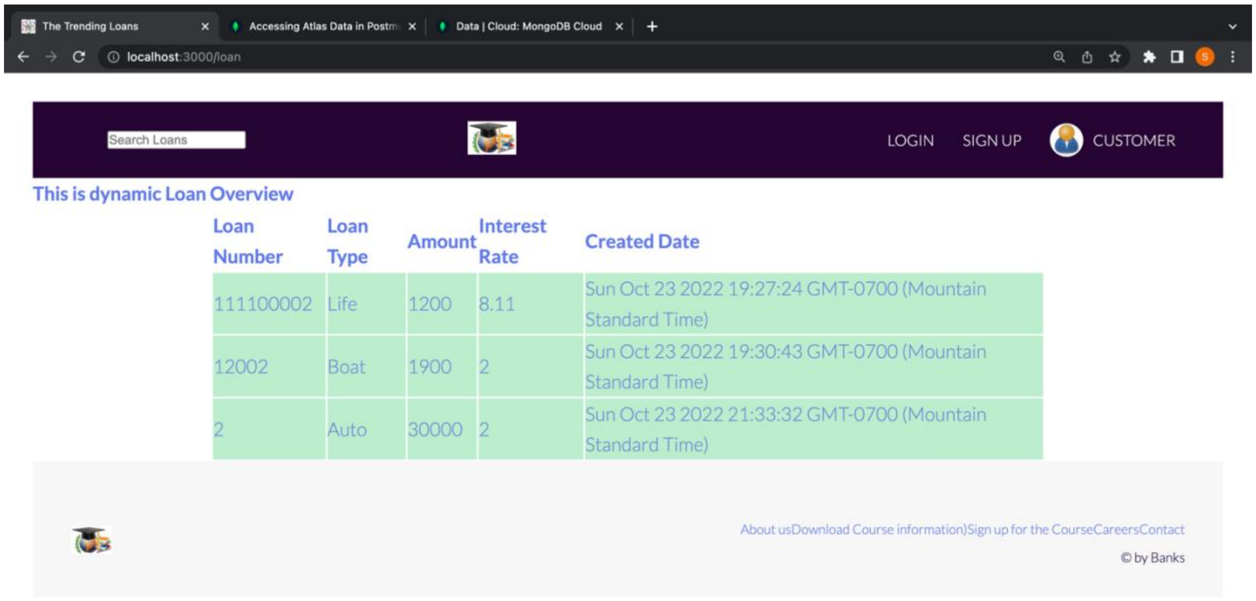
KEY	VALUE	DESCRIPTION
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVC...	
name	test task	

Body — 200 OK 595 ms 339 B

```
{
    "status": "pending",
    "_id": "63670f8cfe27d6d315750176",
    "created_date": "2022-11-06T01:36:12.427Z",
    "__v": 0
}
```

In Module 7: Pages were rendered onto the UI using PUG while retrieving Data from MongoDb:

All Loans Listed:



localhost:3000/loan

This is dynamic Loan Overview

Loan Number	Loan Type	Amount	Interest Rate	Created Date
111100002	Life	1200	8.11	Sun Oct 23 2022 19:27:24 GMT-0700 (Mountain Standard Time)
12002	Boat	1900	2	Sun Oct 23 2022 19:30:43 GMT-0700 (Mountain Standard Time)
2	Auto	30000	2	Sun Oct 23 2022 21:33:32 GMT-0700 (Mountain Standard Time)

About usDownload Course information)Sign up for the CourseCareersContact

© by Banks

In Module 8- Final Project, we protected the routes and only authorized users were allowed to see data pertaining to them which is stored in the DB. The Web Application is Fully Functional and screenshots of the project is at the beginning of the document.

In the Final Project, I was faced with issues related to :

- authorizing a user,
- storing and retrieving Data Related to the User.
- Display a few set of UI elements only if the User is logged In;
- Rendering the views etc.

I was not aware that protecting routes function was implemented in such a way that they retrieve the cookies, create a token, and verify it. I had a hard time verifying a user and authorizing/ allowing them to see only their data. After the issue was resolved, I faced issues with having connection between data collections stored in MongoDb. I was not able to insert the User ID into the Loan Table (to identify a particular loan entry belongs to the user requesting for it). I figured that the token will also contain the User ID and it should be used to store the Loan data in Loan collections for a particular User.

Major Risk was related to not clearing the cookies and being able to access the loan data even after logging out. This was resolved by suitably clearing out the cookies.

Overall, I am aware of protecting the routes, Authenticate and Authorize a User to the Server and retrieve relevant Data from the Database.