Full Stack Development with MERN
Database Design and Development Report

Date	12-07-2024
Team ID	SWTID1720106020
Project Name	Project – Banking Management App
Maximum Marks	5

Project Title: Banking Management App
Date: 12-07-2024
Prepared by: Rangaiah Gari Harshavardhan Reddy, Kopparapu Shashank, Tirumala Mukesh Goud and Nimmala Sai Lahari

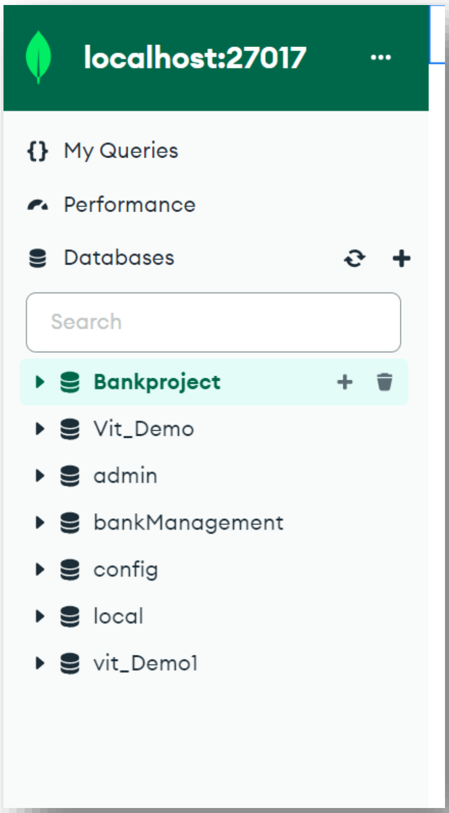
Objective

The objective of this report is to outline the database design and implementation details for the [Your Project Title] project, including schema design and database management system (DBMS) integration.

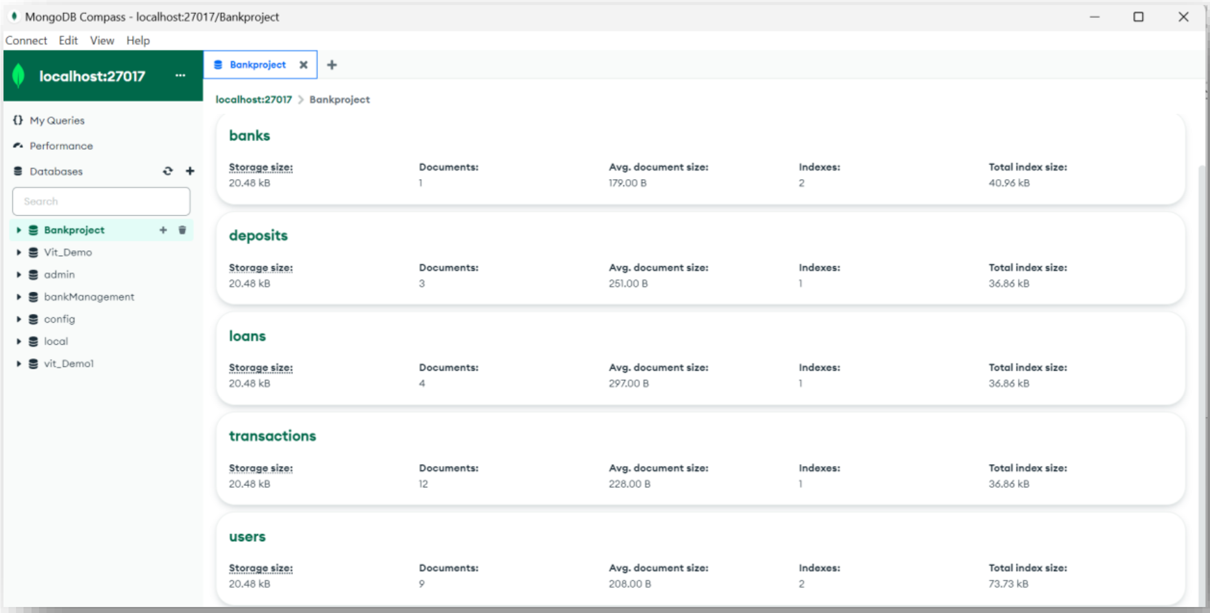
Technologies Used

- **Database Management System (DBMS):** MongoDB
- **Object-Document Mapper (ODM):** Mongoose

Design the Database Schema



<u>MongoDB:</u>



banks

Storage size: 20.48 kB | Documents: 1 | Avg. document size: 179.00 B | Indexes: 2 | Total index size: 40.96 kB

deposits

Storage size: 20.48 kB | Documents: 3 | Avg. document size: 251.00 B | Indexes: 1 | Total index size: 36.86 kB

loans

Storage size: 20.48 kB | Documents: 4 | Avg. document size: 297.00 B | Indexes: 1 | Total index size: 36.86 kB

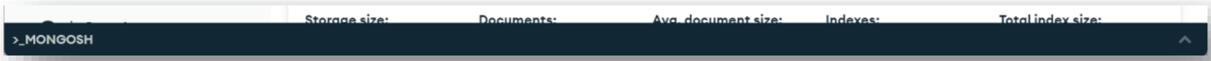
transactions

Storage size: 20.48 kB | Documents: 12 | Avg. document size: 228.00 B | Indexes: 1 | Total index size: 36.86 kB

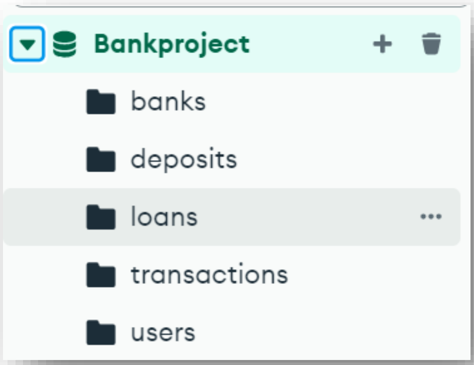
users

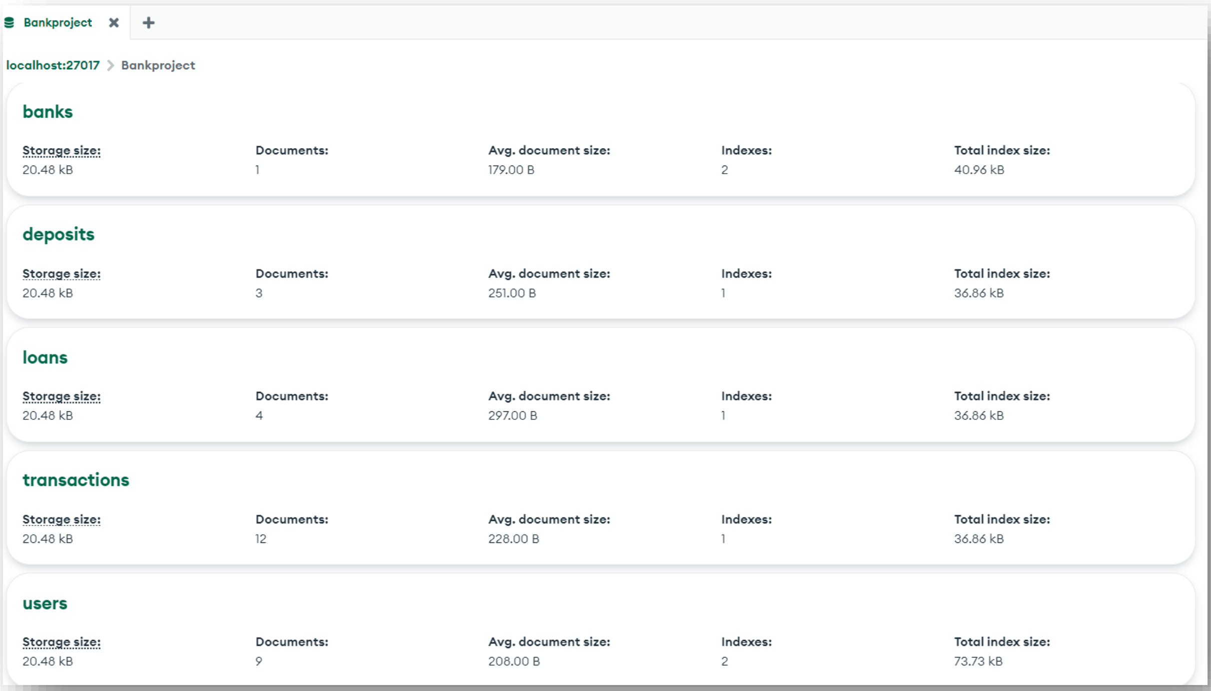
Storage size: 20.48 kB | Documents: 9 | Avg. document size: 208.00 B | Indexes: 2 | Total index size: 73.73 kB

<u>Mongoose:</u>


Storage size: Documents: Avg. document size: Indexes: Total index size:

>_MONGOSH


Bankproject
- banks
- deposits
- loans
- transactions
- users


Bankproject

localhost:27017 > Bankproject

banks

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	1	179.00 B	2	40.96 kB

deposits

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	3	251.00 B	1	36.86 kB

loans

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	4	297.00 B	1	36.86 kB

transactions

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	12	228.00 B	1	36.86 kB

users

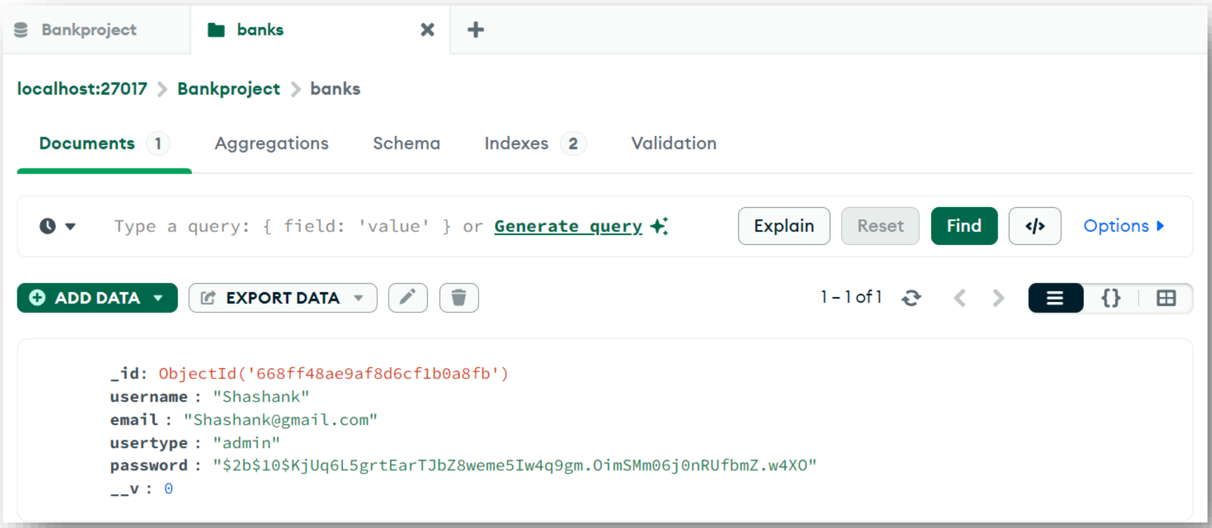
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	9	208.00 B	2	73.73 kB

1. banks

- Attributes:

- username
- email

- usertype
- password

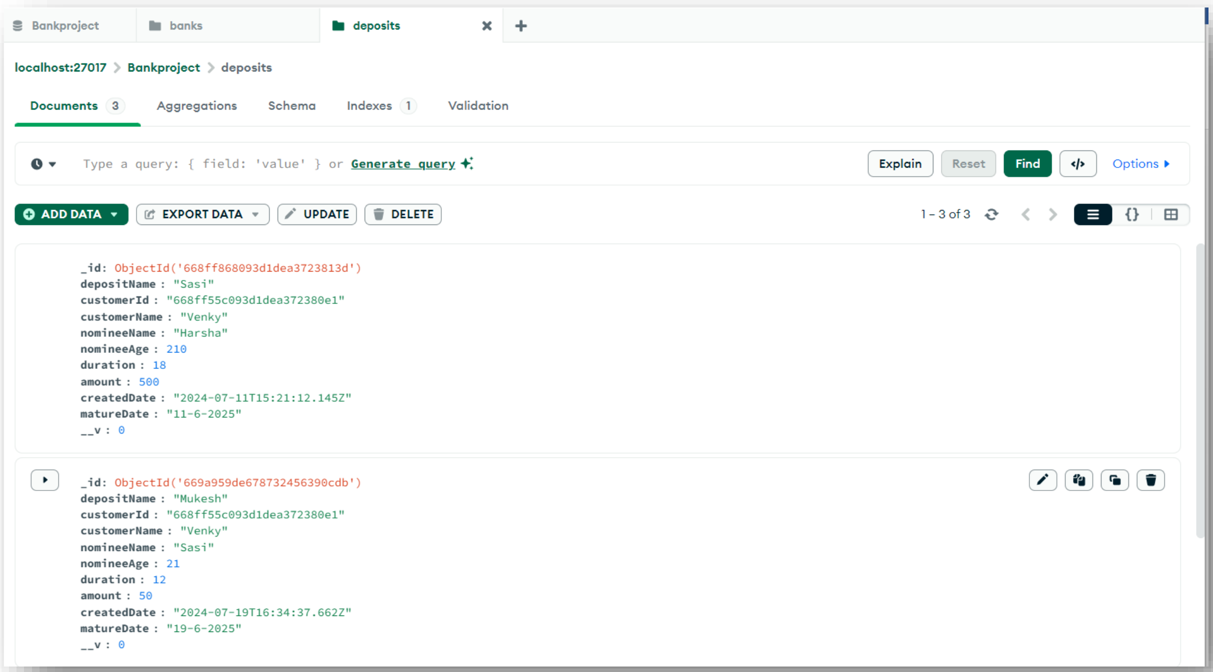


```
_id: ObjectId('668ff48ae9af8d6cf1b0a8fb')
username : "Shashank"
email : "Shashank@gmail.com"
usertype : "admin"
password : "$2b$10$KjUq6L5grtEarTJbZ8weme5Iw4q9gm.OimSMm06j0nRUfbmZ.w4XO"
__v : 0
```

2. deposits

- Attributes:

- _id
- depositName
- customerId
- customerName
- nomineeName
- nomineeAge
- duration
- amount
- createdDate
- matureDate



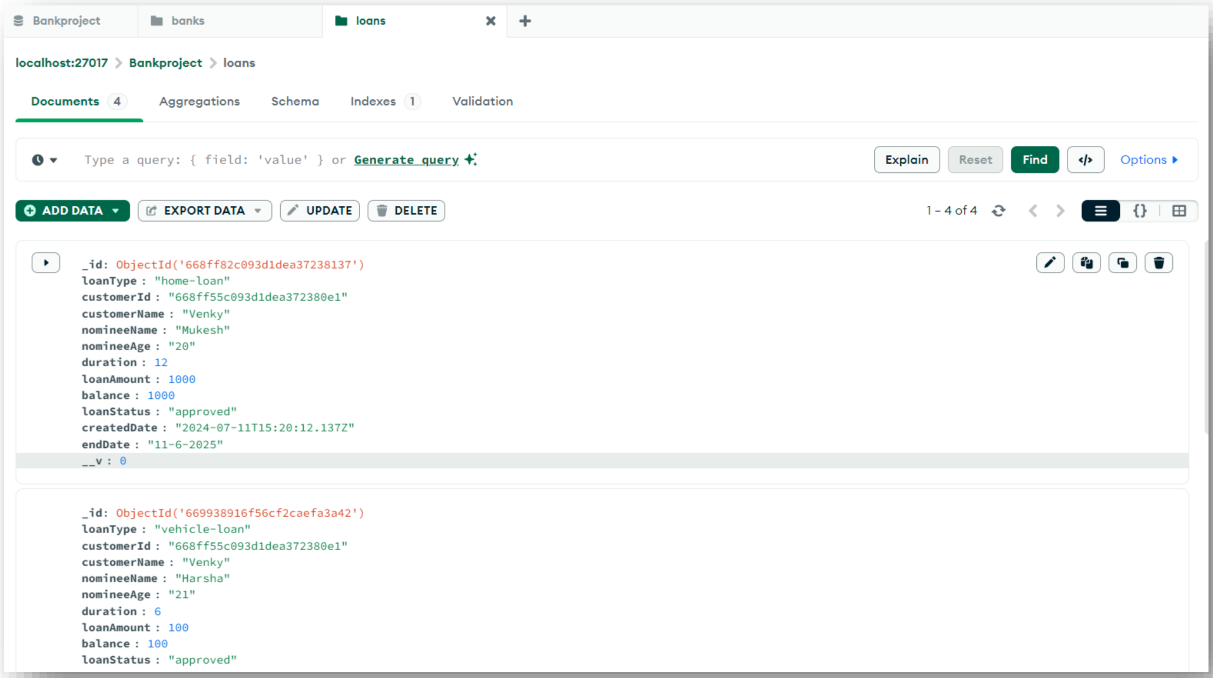
```
_id: ObjectId('668ff868093d1dea3723813d')
depositName : "Sasi"
customerId : "668ff55c093d1dea372380e1"
customerName : "Venky"
nomineeName : "Harsha"
nomineeAge : 210
duration : 18
amount : 500
createdDate : "2024-07-11T15:21:12.145Z"
matureDate : "11-6-2025"
__v : 0
```

```
_id: ObjectId('669a959de678732456390cdb')
depositName : "Mukesh"
customerId : "668ff55c093d1dea372380e1"
customerName : "Venky"
nomineeName : "Sasi"
nomineeAge : 21
duration : 12
amount : 50
createdDate : "2024-07-19T16:34:37.662Z"
matureDate : "19-6-2025"
__v : 0
```

3. loans

- Attributes:

- _id
- loanType
- customerId
- customerName
- nomineeName
- nomineeAge
- duration
- loanAmount
- balance
- loanStatus
- createdDate
- endDate



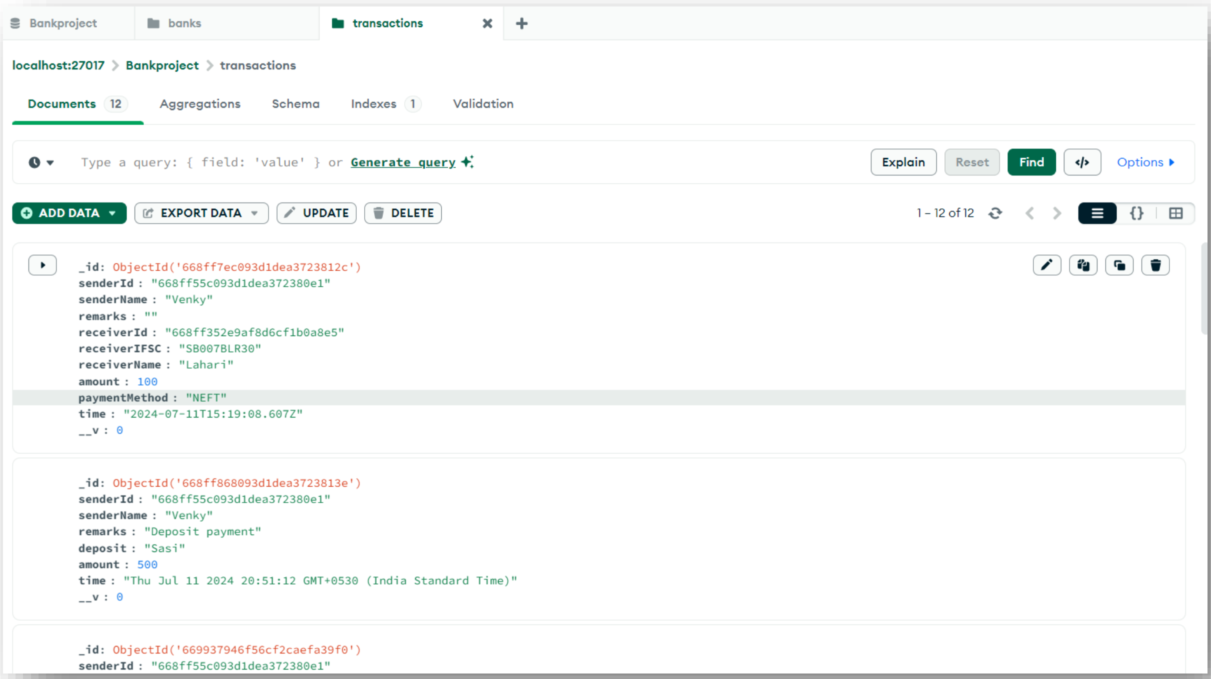
```
_id: ObjectId('668ff82c093d1dea37238137')
loanType : "home-loan"
customerId : "668ff55c093d1dea372380e1"
customerName : "Venky"
nomineeName : "Mukesh"
nomineeAge : "20"
duration : 12
loanAmount : 1000
balance : 1000
loanStatus : "approved"
createdDate : "2024-07-11T15:20:12.137Z"
endDate : "11-6-2025"
__v : 0

_id: ObjectId('669938916f56cf2caefa3a42')
loanType : "vehicle-loan"
customerId : "668ff55c093d1dea372380e1"
customerName : "Venky"
nomineeName : "Harsha"
nomineeAge : "21"
duration : 6
loanAmount : 100
balance : 100
loanStatus : "approved"
```

4. transactions

- Attributes:

- _id
- senderId
- senderName
- remarks
- receiverId
- receiverIFSC
- receiverName
- amount
- paymentMethod
- time



```
_id: ObjectId('668ff7ec093d1dea3723812c')
senderId : "668ff55c093d1dea372380e1"
senderName : "Venky"
remarks : ""
receiverId : "668ff352e9af8d6cf1b0a8e5"
receiverIFSC : "SB007BLR30"
receiverName : "Lahari"
amount : 100
paymentMethod : "NEFT"
time : "2024-07-11T15:19:08.607Z"
__v : 0
```

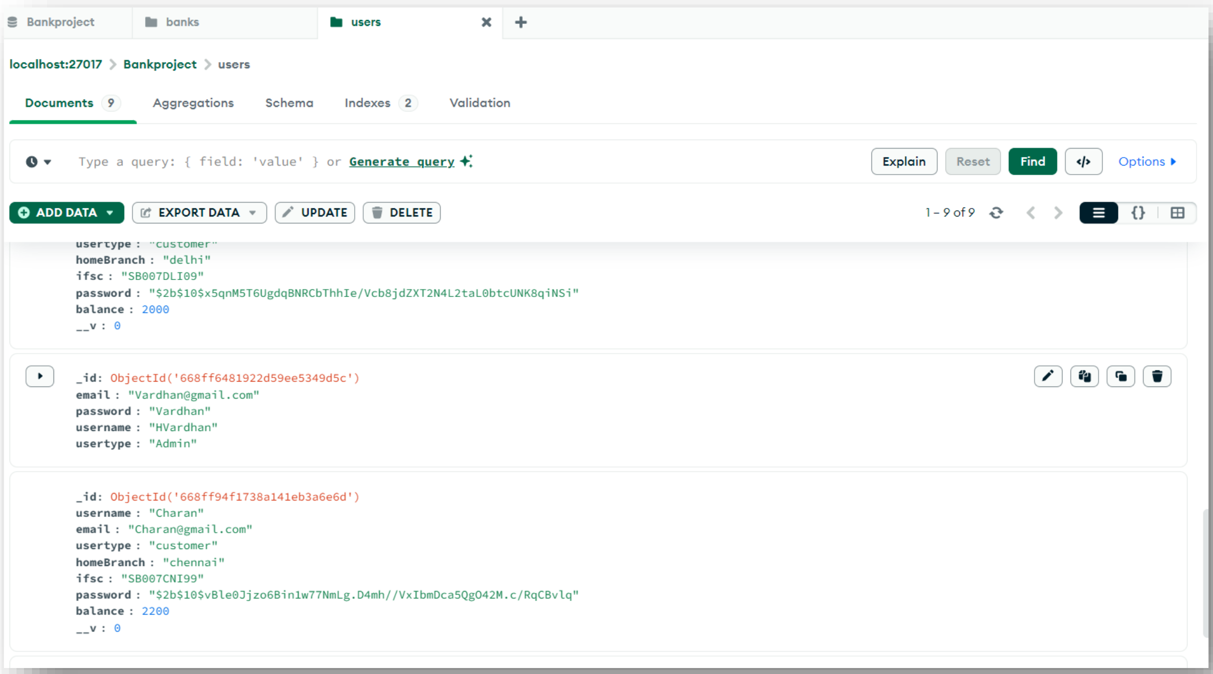
```
_id: ObjectId('668ff868093d1dea3723813e')
senderId : "668ff55c093d1dea372380e1"
senderName : "Venky"
remarks : "Deposit payment"
deposit : "Sasi"
amount : 500
time : "Thu Jul 11 2024 20:51:12 GMT+0530 (India Standard Time)"
__v : 0
```

```
_id: ObjectId('669937946f56cf2caefa39f0')
senderId : "668ff55c093d1dea372380e1"
```

5. users

- Attributes:

- _id
- username
- email
- usertype
- homeBranch
- ifsc
- password
- balance



```
usertype : "customer"
homeBranch : "delhi"
ifsc : "SB007DLI09"
password : "$2b$10$x5qnM5T6UgdqBNRCbThhIe/Vcb8jdZXT2N4L2taL0btcUNK8qiNSi"
balance : 2000
__v : 0

_id: ObjectId('668ff6481922d59ee5349d5c')
email : "Vardhan@gmail.com"
password : "Vardhan"
username : "HVardhan"
usertype : "Admin"

_id: ObjectId('668ff94f1738a141eb3a6e6d')
username : "Charan"
email : "Charan@gmail.com"
usertype : "customer"
homeBranch : "chennai"
ifsc : "SB007CNI99"
password : "$2b$10$vBle0Jjzo6Bin1w77NmLg.D4mh//VxIbmDca5Qg042M.c/RqCBvlq"
balance : 2200
__v : 0
```

Implement the Database using MongoDB

The MongoDB is implemented by the following connections and structures:

Database Name: **Bankproject**

1.collections: banks

- schema:

```
{
  "_id": {
    "$oid": "668ff35e1922d59ee5349d5b"
  },
  "password": "Sasi",
  "username": "Shashank",
  "usertype": "Admin",
  "email": "Shashank@gmail.com"
}
```

2.collections: deposits

- schema:

```
{
  "_id": {
    "$oid": "668ff868093d1dea3723813d"
  },
  "depositName": "Sasi",
  "customerId": "668ff55c093d1dea372380e1",
  "customerName": "Venky",
  "nomineeName": "Harsha",
  "nomineeAge": 210,
  "duration": 18,
  "amount": 500,
  "createdDate": "2024-07-11T15:21:12.145Z",
  "matureDate": "11-6-2025",
  "__v": 0
}
```

3.collections: loans

   - schema:

```
{
  "_id": {
    "$oid": "668ff82c093d1dea37238137"
  },
  "loanType": "home-loan",
  "customerId": "668ff55c093d1dea372380e1",
  "customerName": "Venky",
  "nomineeName": "Mukesh",
  "nomineeAge": "20",
  "duration": 12,
  "loanAmount": 1000,
  "balance": 1000,
  "loanStatus": "approved",
  "createdDate": "2024-07-11T15:20:12.137Z",
  "endDate": "11-6-2025",
  "__v": 0
}
```

4.collections: transactions

- schema:

```
{
  "_id": {
    "$oid": "669a9e2d8320a6f196e2bcfa"
  },
  "senderId": "669a9b6010c0dc302bc7f5f3",
  "remarks": "Loan re-payment",
  "loan": "vehicle-loan",
  "amount": 50,
  "time": "Fri Jul 19 2024 22:41:09 GMT+0530 (India Standard Time)",
  "__v": 0
}
```

———

```
{
  "_id": {
    "$oid": "668ff7ec093d1dea3723812c"
  },
  "senderId": "668ff55c093d1dea372380e1",
  "senderName": "Venky",
  "remarks": "",
  "receiverId": "668ff352e9af8d6cf1b0a8e5",
  "receiverIFSC": "SB007BLR30",
  "receiverName": "Lahari",
  "amount": 100,
  "paymentMethod": "NEFT",
  "time": "2024-07-11T15:19:08.607Z",
  "__v": 0
}
```

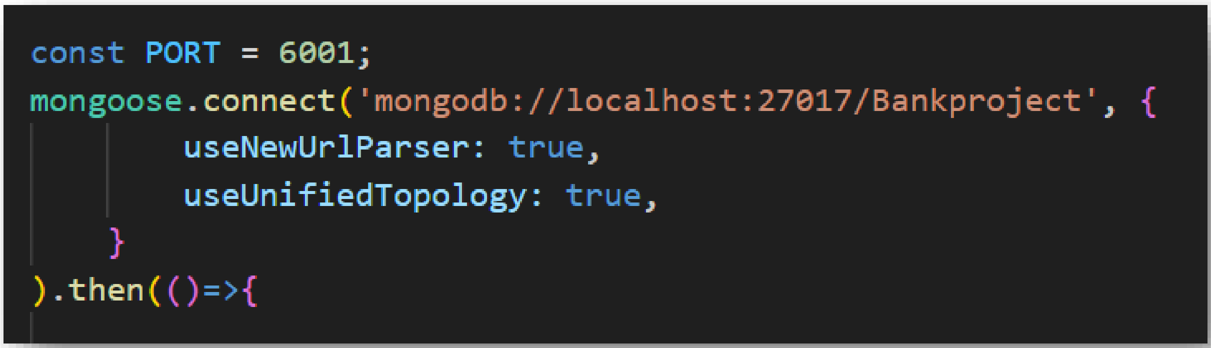
5.collections: users

- schema:

```
{
  "_id": {
    "$oid": "668ff5ef093d1dea372380ef"
  },
  "username": "Sasi",
  "email": "Sasi@gmail.com",
  "usertype": "customer",
  "homeBranch": "tirupati",
  "ifsc": "SB007TPTY05",
  "password": "$2b$10$pP00iqsD6S5phKlHqVx1NegAS5eG..60uxdS6Gb0OtVKEMvNpHQrG",
  "balance": 2000,
  "__v": 0
}
```

Integration with Backend

- Database connection: Give Screenshot of Database connection done using Mongoose



```
const PORT = 6001;
mongoose.connect('mongodb://localhost:27017/Bankproject', {
        useNewUrlParser: true,
        useUnifiedTopology: true,
    }
).then(()=>{
```

- The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:

User Management

User Management is a crucial aspect of the application, enabling comprehensive CRUD (Create, Read, Update, Delete) operations for users. Through the use of endpoints such as POST /register, the application allows new users to register by saving their details, including username, email, password, and user type (either admin or customer), into a database like MongoDB. Once registered, user details can be retrieved via the GET /user-details/:id endpoint, updated using the PUT /update-user/:id endpoint, and deleted with the DELETE /delete-user/:id endpoint. These operations ensure that user data is meticulously maintained and easily accessible, contributing to a seamless user experience.

User Registration

The registration process involves the creation of a new user account. When a user submits their details through the POST /register endpoint, the application validates the input data and stores it in the database. This ensures that only valid users can access the application, maintaining security and integrity.

User Retrieval

The GET /user-details/:id endpoint allows users to retrieve their own details or those of another user by specifying the user ID. This endpoint is useful for users who need to view or update their own profiles or for administrators who need to manage user accounts.

User Update

The PUT /update-user/:id endpoint enables users to update their own details, such as changing their username, email, or password. This endpoint ensures that users can easily manage their own profiles and keep their information up to date.

User Deletion

The DELETE /delete-user/:id endpoint allows users to delete their own accounts, removing all associated data from the database. This endpoint is useful for users who no longer wish to use the application or need to terminate their account for any reason.

Post Management

Post Management is another essential component, permitting users to create, read, update, and delete posts, all while ensuring user authentication to maintain security. Authenticated users can create new posts via the POST /create-post endpoint, which stores post details like title, content, author ID, and creation date in the database. The GET /posts endpoint allows for the retrieval of all posts, enabling their display within the application. Existing posts can be updated using the PUT /update-post/:id endpoint, and deleted through the DELETE /delete-post/:id endpoint. This comprehensive post management system ensures that the content within the application is current, accurate, and secure.

Post Creation

The POST /create-post endpoint allows users to create new posts by submitting the post details, such as title, content, and author ID. The application validates the input data and stores the post in the database, ensuring that only valid posts are created.

Post Retrieval

The GET /posts endpoint retrieves all posts from the database, allowing them to be displayed within the application. This endpoint is useful for users who want to view all available posts or for administrators who need to manage the content.

Post Update

The PUT /update-post/:id endpoint enables users to update existing posts, such as changing the title, content, or author. This endpoint ensures that posts can be easily updated and kept current.

Post Deletion

The DELETE /delete-post/:id endpoint allows users to delete existing posts, removing them from the database. This endpoint is useful for users who no longer wish to display a post or need to remove it for any reason.

Comment Management

Comment Management further enhances user interaction by allowing users to add, read, update, and delete comments on posts. This feature is vital for fostering user engagement and dialogue within the application. Users can add comments to posts via the POST /create-comment endpoint, which records comment details such as post ID, user ID, content, and creation date in the database. The GET /comments/:postId endpoint retrieves all comments associated with a specific post, facilitating their display. Users can update their comments using the PUT /update-comment/:id endpoint and delete them with the DELETE /delete-comment/:id endpoint. These capabilities ensure that comments are accurately and securely managed, enhancing the overall user experience.

Comment Creation

The POST /create-comment endpoint allows users to add new comments to posts by submitting the comment details, such as post ID, user ID, and content. The application validates the input data and stores the comment in the database, ensuring that only valid comments are created.

Comment Retrieval

The GET /comments/:postId endpoint retrieves all comments associated with a specific post, allowing them to be displayed within the application. This endpoint is useful for users who want to view all comments for a particular post or for administrators who need to manage comments.

Comment Update

The PUT /update-comment/:id endpoint enables users to update existing comments, such as changing the content or user ID. This endpoint ensures that comments can be easily updated and kept current.

Comment Deletion

The DELETE /delete-comment/:id endpoint allows users to delete existing comments, removing them from the database. This endpoint is useful for users who no longer wish to display a comment or need to remove it for any reason.

By providing comprehensive CRUD operations for users, posts, and comments, the application ensures that all data is accurately and securely managed, contributing to a seamless user experience.