Full Stack Development with MERN

Database Design and Development Report

Date	09-07-2024
Team ID	SWTID1720104852
Project Name	Banking Management App(MERN)
Maximum Marks	5

Project Title: Banking Management

Date: 09-07-2024

Prepared by: Hithesh S

Team Members:

Chandra Kishore Sure

Venkata Krishna C

Venkata Subrahmanya Deepak N

Objective

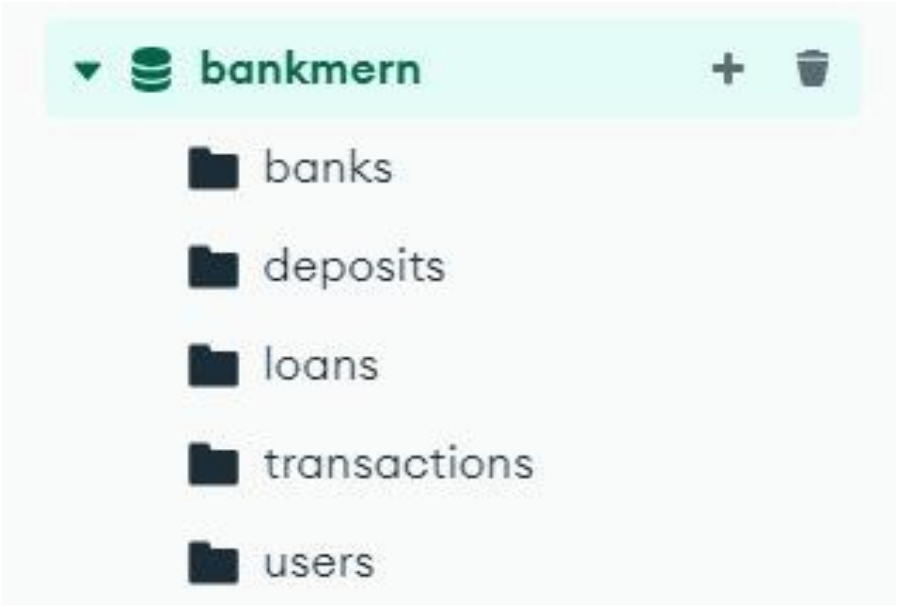
The objective of this report is to outline the database design and implementation details for the Banking Management project, including schema design and database management system (DBMS) integration.

Technologies Used

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

Design the Database Schema

The database schema is designed to accommodate the following entities and relationships:



banks

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	1	177.00 B	2	40.96 kB

deposits

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	1	257.00 B	1	20.48 kB

loans

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	3	301.00 B	1	36.86 kB

transactions

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	5	248.00 B	1	36.86 kB

users

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	3	202.00 B	2	73.73 kB

1. banks

- Attributes:

- username
- email
- usertype
- password

2. deposits

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

3. loans

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

4. transactions

- Attributes:
    - _id
    - senderId

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

5. users

- Attributes:

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

Implement the Database using MongoDB

The MongoDB database is implemented with the following collections and structures:

Database Name: bankmern

1. Collection: banks

- Schema:

```
{
  "username": "kishore",
```

```
  "email": "kishore@gmail.com",
  "usertype": "admin",
  "password": "$2b$10$eStjvj4cYr5DbQwtEkFkVu1Rpg9ZPoSl3fLYhjXdR8VCie8reEYZe",
  "__v": 0
}
```


2. Collection: deposits

- Schema:


```
{
  "_id": {
    "$oid": "668ff0d5d8eeeb60a5f4a601"
  },
  "depositName": "Hithesh",
  "customerId": "668fcda0499862ccb96e173f",
  "customerName": "Hithesh",
  "nomineeName": "venkat",
  "nomineeAge": 35,
  "duration": 12,
  "amount": 1000,
  "createdDate": "2024-07-11T14:48:53.164Z",
  "matureDate": "11-6-2025",
  "__v": 0
}
```


3. Collection: loans

- Schema:

```
```

```
{
  "_id": {
    "$oid": "668fdf7c1105a3ae86a39049"
  },
  "loanType": "home-loan",
  "customerId": "668fcda0499862ccb96e173f",
  "customerName": "Hithesh",
  "nomineeName": "hithesh",
  "nomineeAge": "40",
  "duration": 6,
  "loanAmount": 1000,
  "balance": 1000,
  "loanStatus": "approved",
  "createdDate": "2024-07-11T13:34:52.865Z",
  "endDate": "11-6-2024",
  "__v": 0
}
```


4. Collection: transactions

- Schema:


```
{
  "_id": {
    "$oid": "668fe06cc3fac233481de158"
  },
  "senderId": "668fcda0499862ccb96e173f",
  "senderName": "Hithesh",
  "remarks": "",
  "receiverId": "668fe01cc3fac233481de14b",
  "receiverIFSC": "SB007BLR30",
```

```
  "receiverName": "venkat",
  "amount": 10,
  "paymentMethod": "NEFT",
  "time": "2024-07-11T13:38:52.128Z",
  "__v": 0
}
```

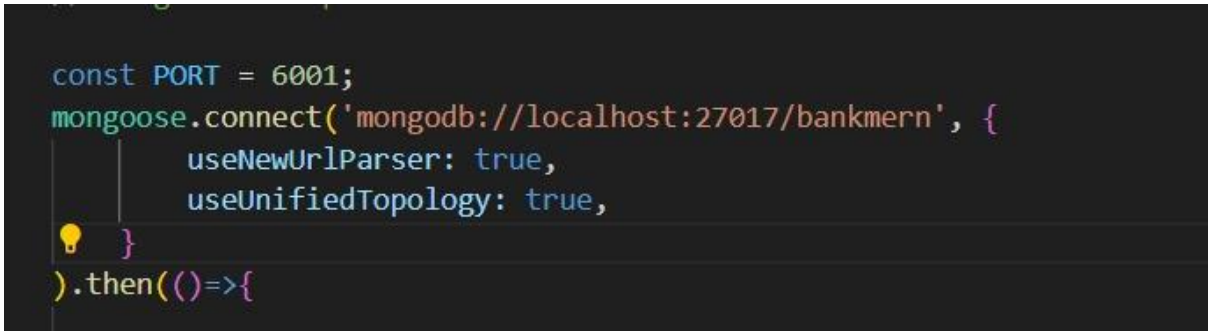

5. Collection: users

- Schema:


```
{
  "_id": {
    "$oid": "668fcda0499862ccb96e173f"
  },
  "username": "Hithesh",
  "email": "sivaramhithesh191009410@gmail.com",
  "usertype": "customer",
  "homeBranch": "hyderabad",
  "ifsc": "SB007HYD25",
  "password": "$2b$10$0i.Myx/Rixkox7bl9e2X7.VEShVIT166OzSytGih6kFyInQ.9SBee",
  "balance": 1690,
  "__v": 0
}
```


Integration with Backend

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



- The backend APIs interact with MongoDB using Mongoose ODM Key interactions include:
    - User Management: CRUD operations for users.

      User Management is fundamental to 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 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.

    - Post Management: CRUD operations for posts, with user authentication.

      Post Management is another crucial 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 MongoDB. 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.

    - Comment Management: CRUD operations for comments associated with posts.

      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 MongoDB. 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 well-managed and relevant, promoting a dynamic and interactive user environment.