MongoDB Project

28.04.2023

Group Number:

Name	Roll Number	Group
Swapnil	102016108	3CS12
Aastha Chhabra	102016054	3CS10

Overview

MongoDB Note-Taking WebApp using ExpressJS, MongoDB, and EJS

Introduction

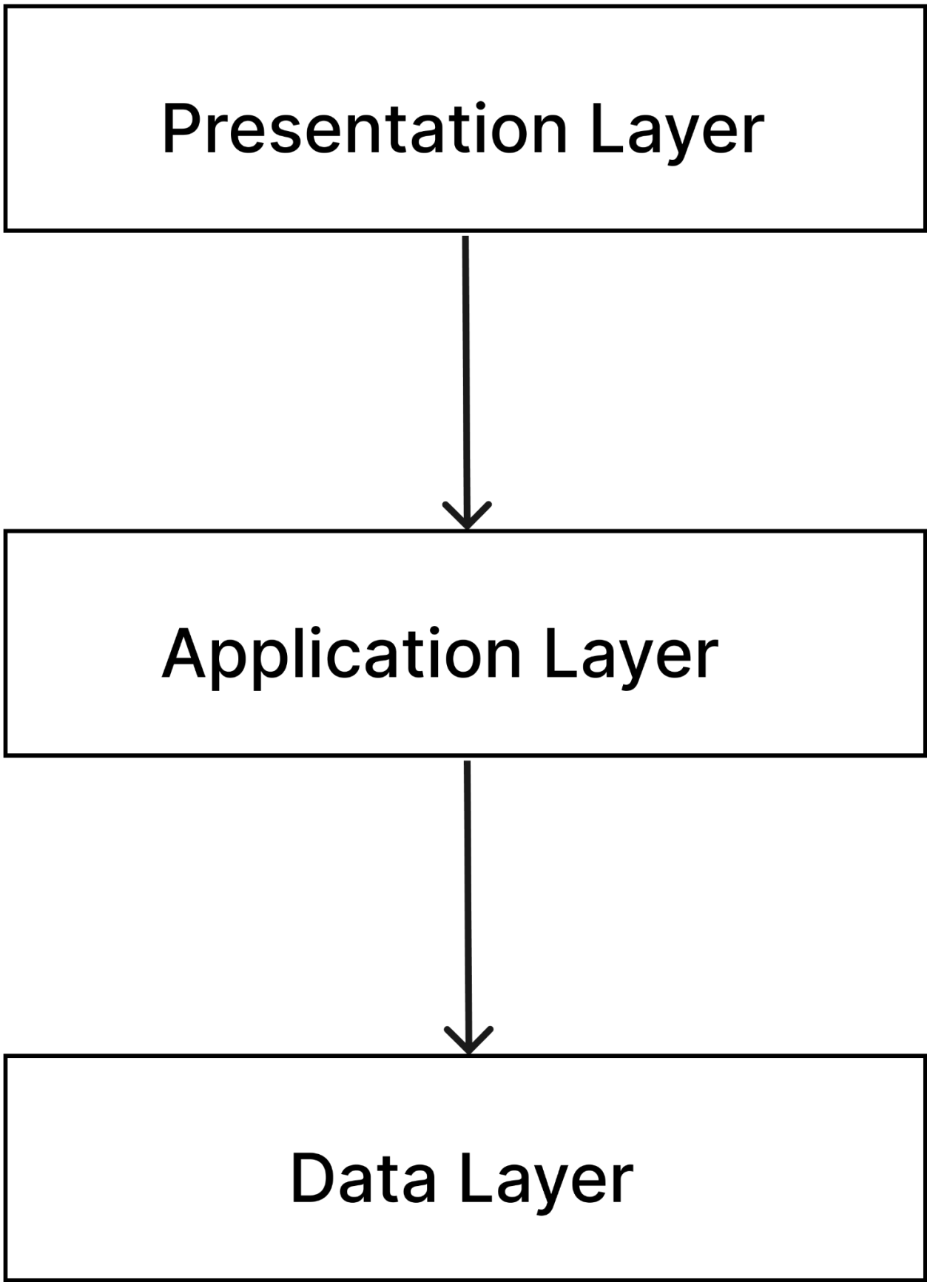
The purpose of this project report is to provide an overview of a note-taking web application built using MongoDB as the database, ExpressJS as the backend framework, and EJS as the templating engine. This web application provides users with the ability to create, read, update, and delete notes, as well as search for notes based on their content.

Methodology

The web application was built using the Model-View-Controller (MVC) architectural pattern. The model component was implemented using MongoDB, which provided a schemaless, document-based approach to storing data. The controller component was implemented using ExpressJS, which provided a robust framework for building RESTful APIs. The view component was implemented using EJS, which allowed for the dynamic generation of HTML pages based on data from the database.

System Architecture

The system architecture of the MongoDB Note-Taking web application using ExpressJS, MongoDB, and EJS can be divided into three main layers: the presentation layer, the application layer, and the data layer. The following diagram illustrates the high-level system architecture of the application:


Presentation Layer
Application Layer
Data Layer

1. **Presentation Layer:** The presentation layer is responsible for the user interface of the application. It is built using HTML, CSS, and JavaScript, and it is rendered using the EJS templating engine. The presentation layer handles user input and displays the results of user actions. It communicates with the application layer via HTTP requests.
2. **Application Layer:** The application layer is responsible for the business logic of the application. It handles user requests, processes them, and communicates with the data layer to retrieve or update data as needed. The application layer is built using ExpressJS, which provides an easy-to-use and flexible framework for building web applications. It communicates with the presentation layer via HTTP requests and with the data layer via MongoDB queries.
3. **Data Layer:** The data layer is responsible for storing and retrieving data used by the application. It is built using MongoDB, a NoSQL database that provides a flexible and scalable data storage solution. The data layer communicates with the application layer via MongoDB queries and with the presentation layer indirectly through the application layer.

Overall, the system architecture of the MongoDB Note-Taking web application is designed to be scalable, flexible, and maintainable. The separation of concerns between the presentation, application, and data layers allows for easy modification and extension of the application as needed.

Database Design

The MongoDB Note-Taking web application using ExpressJS, MongoDB, and EJS uses three collections in the database: users, notes, and sessions. The following is a description of the schema design for each collection:

1. **Users Collection:** The users collection stores information about the users of the application. Each document in the users collection contains the following fields:

- _id: A unique identifier for the user document.

- googleId: The Google ID of the user, if the user signed up using their Google account.
- displayName: The display name of the user.
- firstName: The first name of the user.
- lastName: The last name of the user.
- profileImage: The URL of the user's profile image.
- createdAt: The timestamp when the user document was created.

The username and email fields are not included, as they are not needed if the user signed up using their Google account.

2. **Notes Collection:** The notes collection stores information about the notes created by the users. Each document in the notes collection contains the following fields:

- _id: A unique identifier for the note document.
- user: The ID of the user who created the note.
- title: The title of the note.
- body: The content of the note.
- createdAt: The timestamp when the note document was created.
- updatedAt: The timestamp when the note document was last updated.

The created_by field is renamed to user, and it is a reference to the _id field of a user document in the users collection, indicating which user created the note. This enables the application to retrieve all notes created by a specific user.

3. **Sessions Collection:** The sessions collection stores information about the user sessions. Each document in the sessions collection contains the following fields:

- _id: A unique identifier for the session document.
- session: The session data.

- expires: The timestamp when the session expires.
- user: The ID of the user associated with the session.

The session_id field is renamed to _id, and the user_id field is included as a reference to the _id field of a user document in the users collection, indicating which user the session is associated with.

Overall, the revised database design of the MongoDB Note-Taking web application is designed to include the additional fields for the users collection and to use appropriate references between collections to enable fast and easy retrieval of data.

Features

The note-taking web application provides the following features:

- User authentication using PassportJS
- Creation of notes with a title and content
- Fetching of notes from the database
- Updating of notes
- Deletion of notes
- Searching for notes based on content
- Pagination of notes
- Sorting the notes based on the updated time

Technologies used

- MongoDB: A NoSQL database that provides a schemaless, document-based approach to storing data.
- ExpressJS: A backend web framework for Node.js that provides a robust set of features for building RESTful APIs.
- EJS: A templating engine for Node.js that allows for the dynamic generation of HTML pages based on data from the database.

- PassportJS: An authentication middleware for Node.js that provides a flexible and modular approach to user authentication.

Implementation

The implementation of the MongoDB Note-Taking web application involved the following steps:

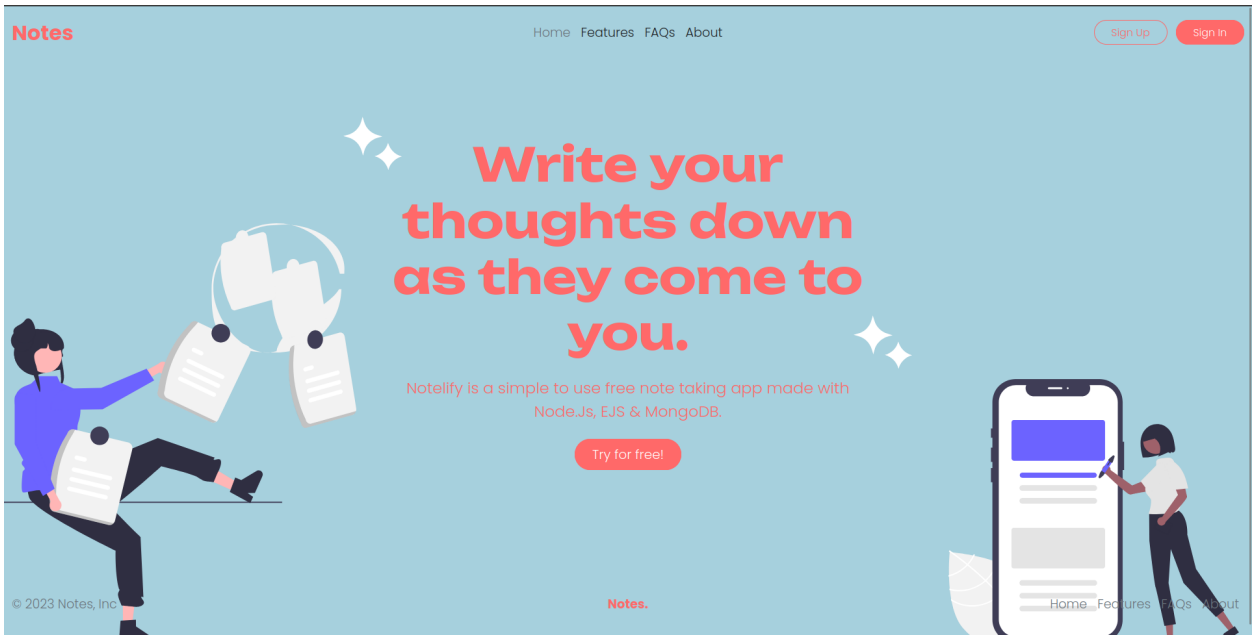
1. Installed and configured MongoDB, ExpressJS, and EJS using the appropriate package managers.
2. Set up the MongoDB database and collections with the appropriate fields, as per the database design outlined above.
3. Configured the ExpressJS server to handle HTTP requests from the client-side.
4. Defined routes for the various CRUD operations on the notes collection, along with their respective controller functions that interact with the database.
5. Implemented authentication and authorization using the Passport.js middleware, using the users collection to store user data and the sessions collection to manage user sessions.
6. Used EJS templates to render dynamic views based on the data retrieved from the MongoDB database.
7. Tested the application using tools such as Postman or a web browser, ensuring that all CRUD operations and authentication and authorization functionality worked as expected.

Conclusion

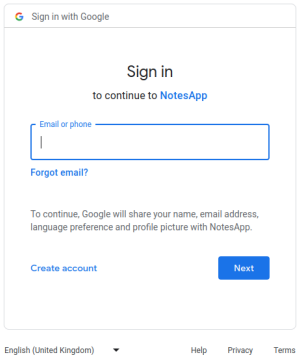
In conclusion, the Note-taking web application built using MongoDB, ExpressJS, and EJS provides users with a robust and scalable solution for managing their notes. The use of MongoDB allows for a flexible and scalable approach to storing data, while the use of ExpressJS and EJS provides a powerful and easy-to-use platform for building and rendering web pages. Overall, this project demonstrates the power and flexibility of using MongoDB, ExpressJS, and EJS for building web applications.

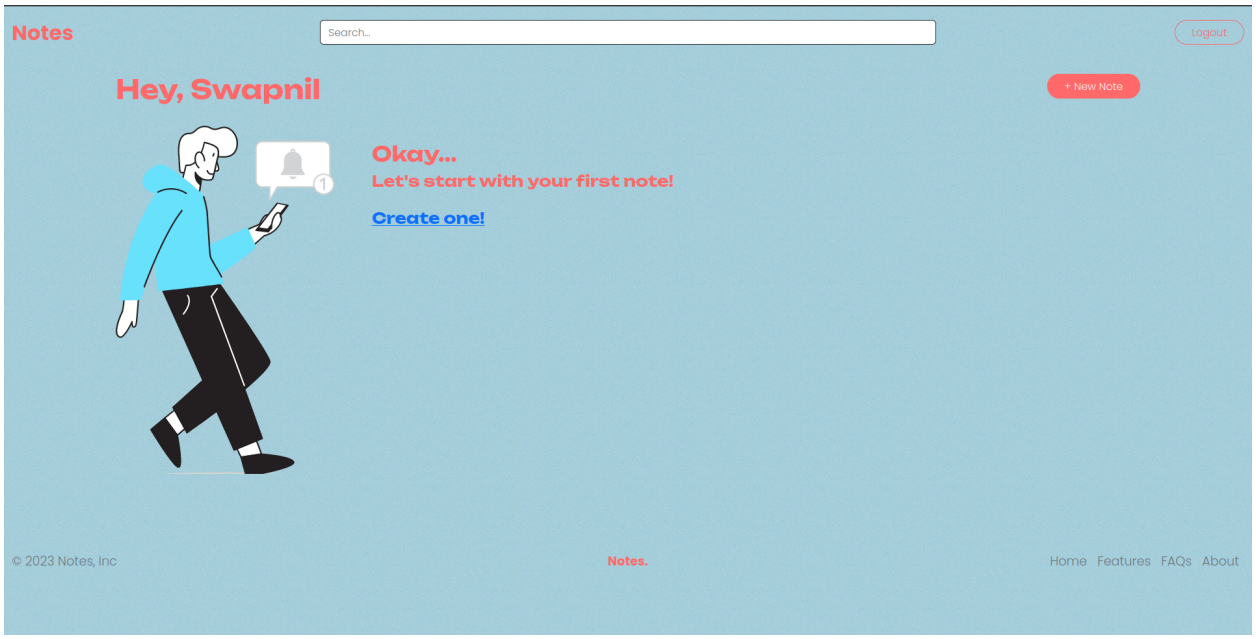
Output Screenshots

Home Page:

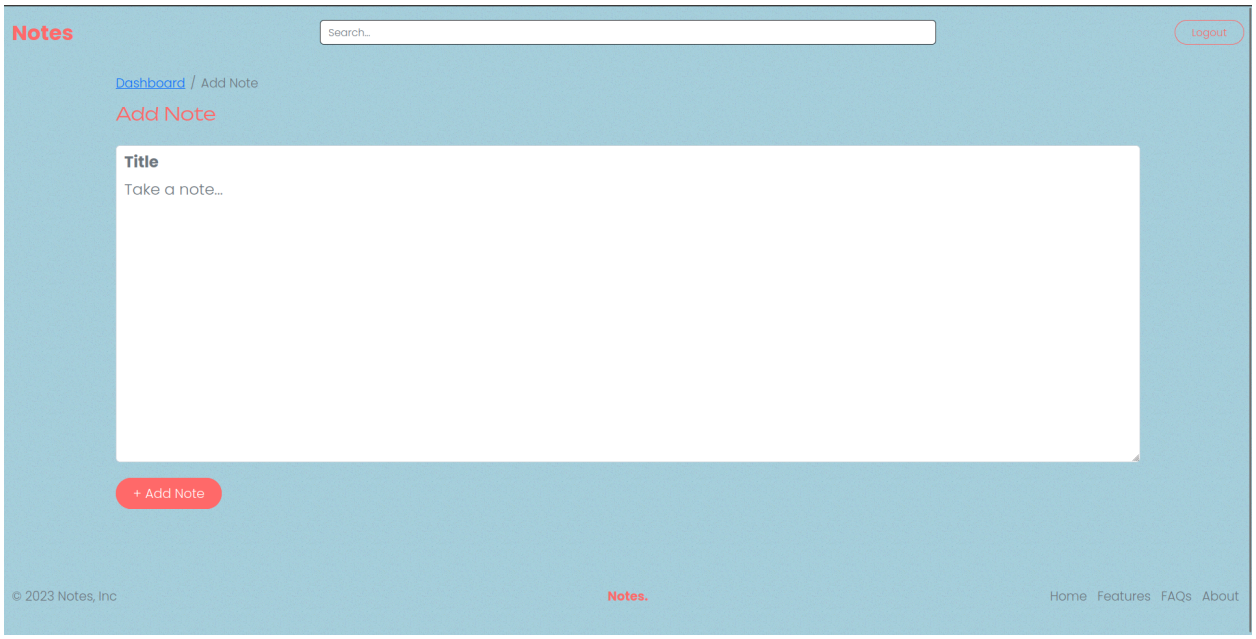


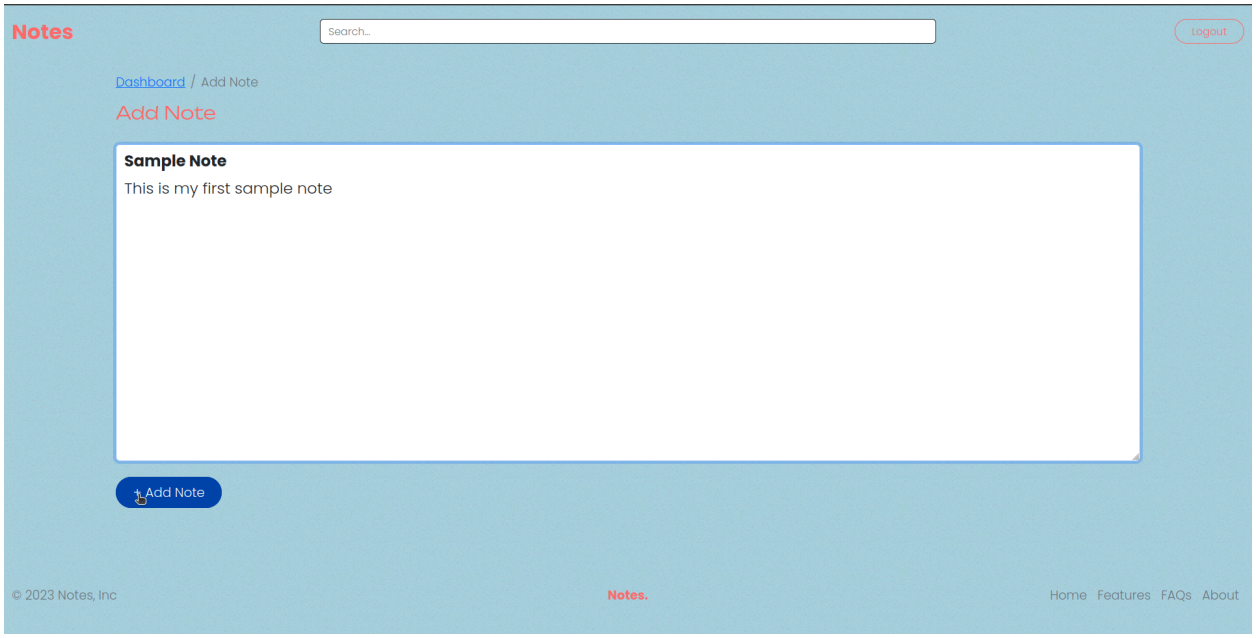
Signup/Signin Page:



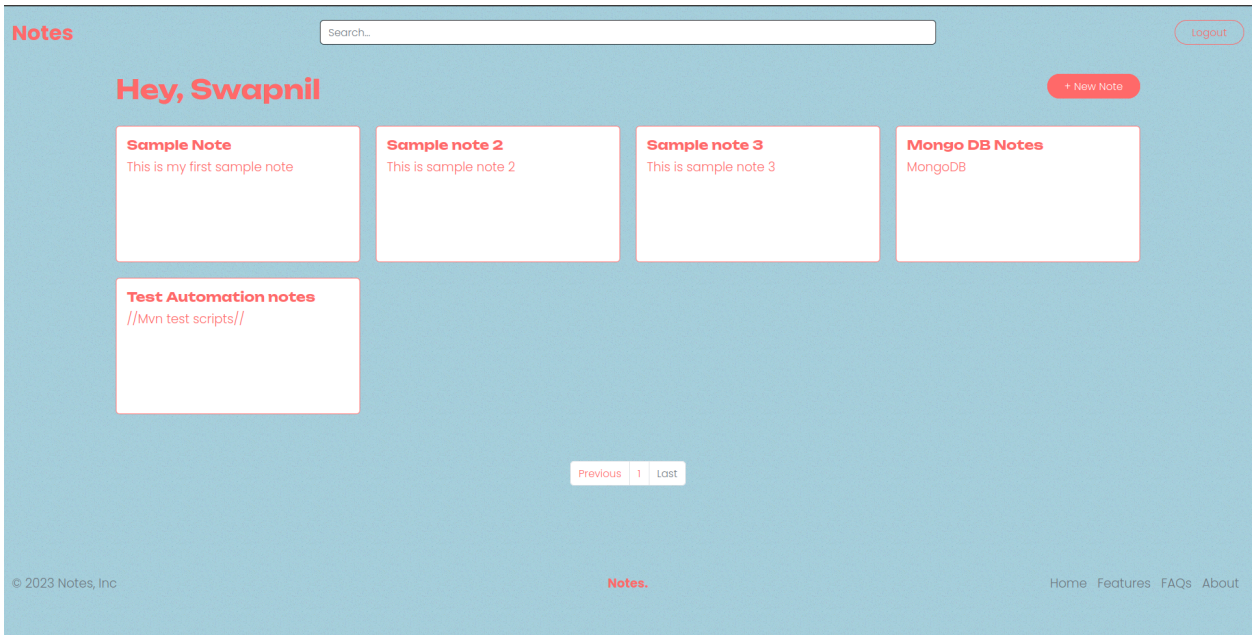


New Note Page:

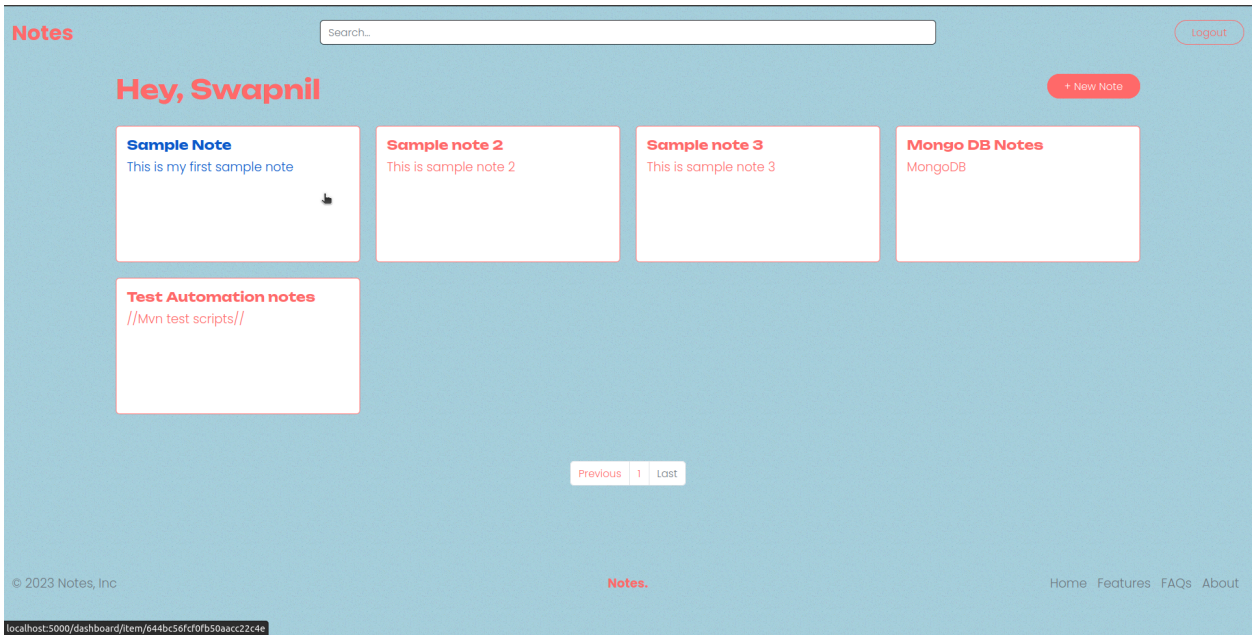




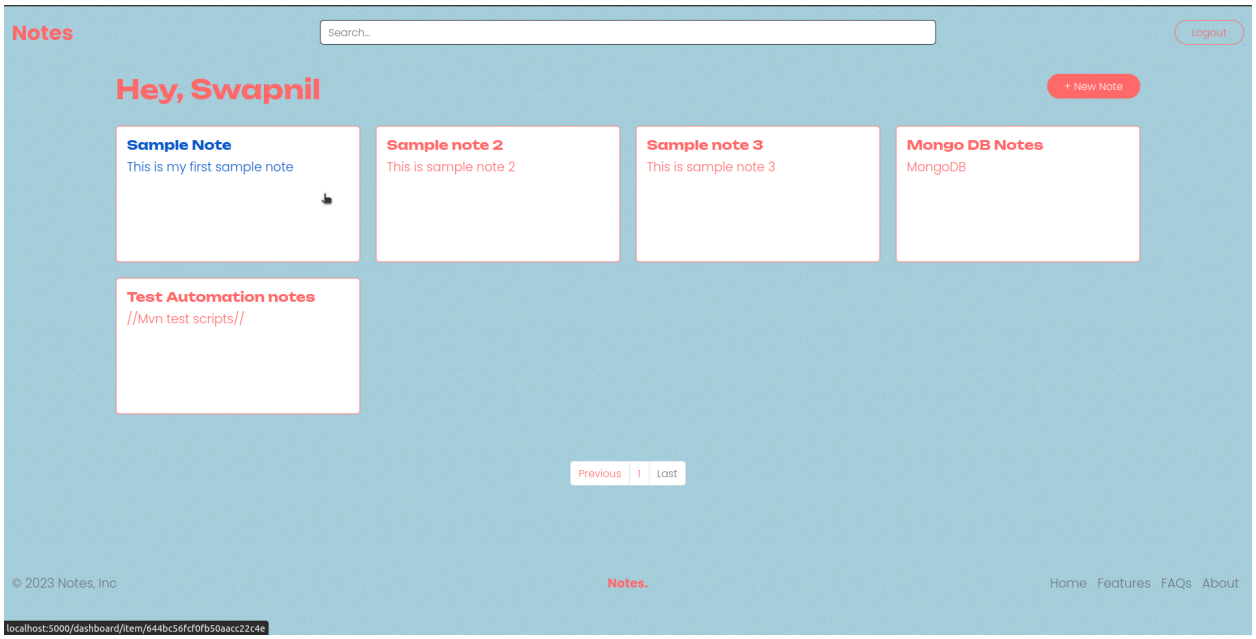
Dashboard (After adding multiple notes)

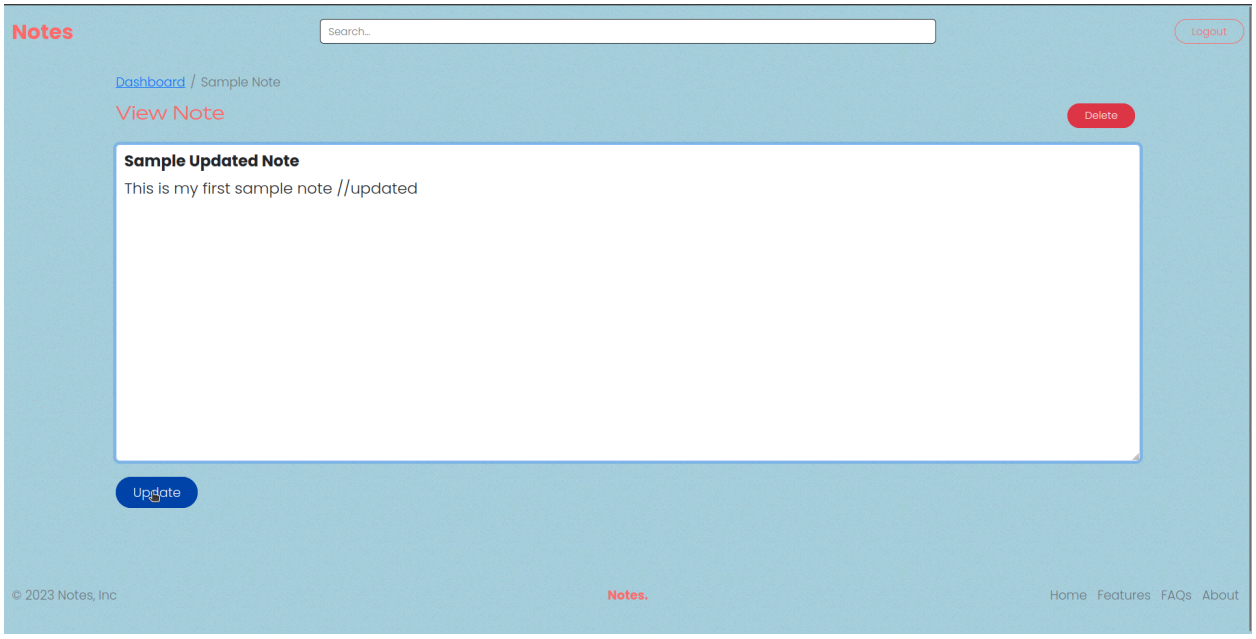


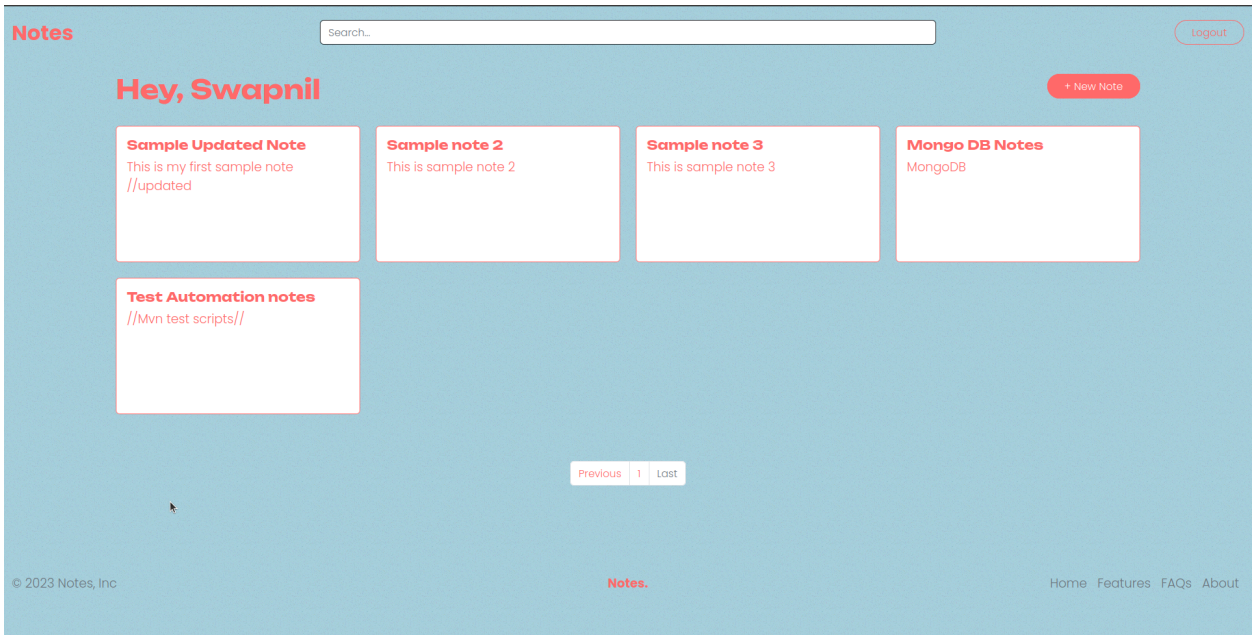
Reading Notes (After adding multiple notes in database)



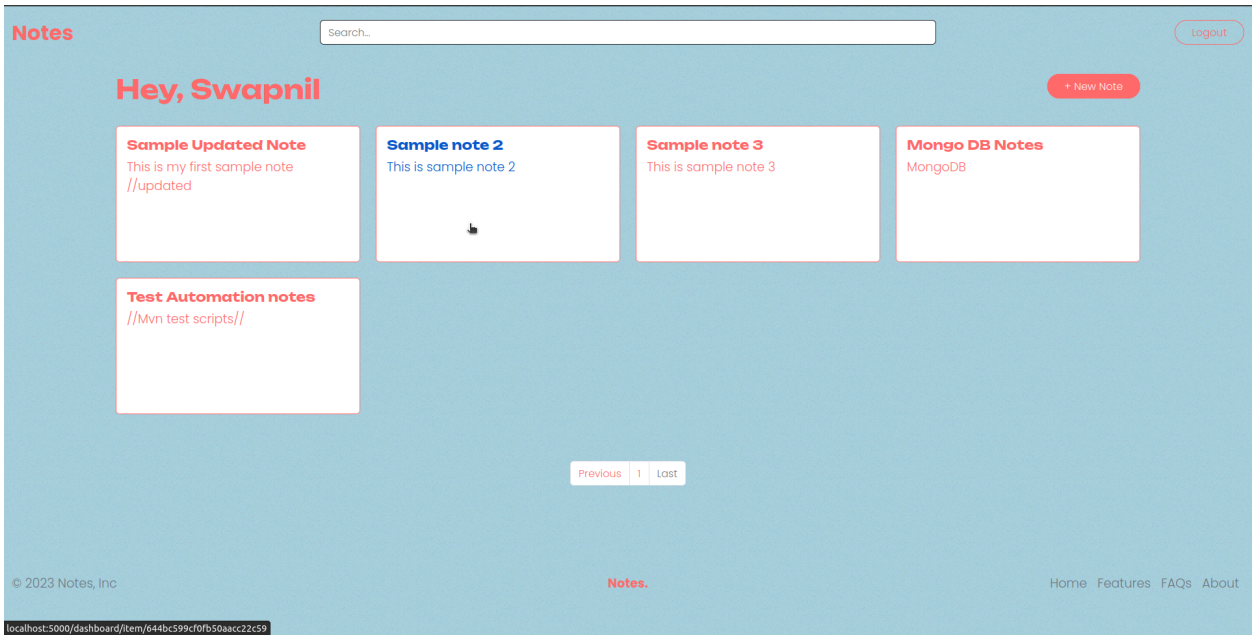
Updating Notes

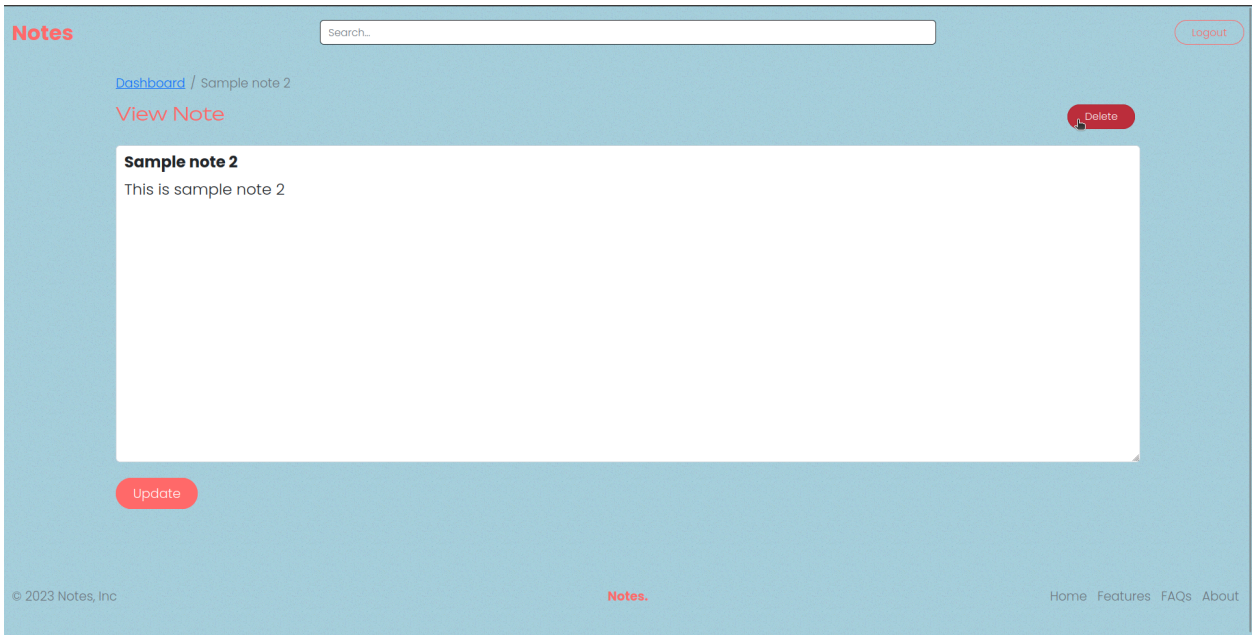

Notes
Search..
Logout
Hey, Swapnil
+ New Note
Sample Note
This is my first sample note
Sample note 2
This is sample note 2
Sample note 3
This is sample note 3
Mongo DB Notes
MongoDB
Test Automation notes
//Mvn test scripts//
Previous
1
Last
© 2023 Notes, Inc
Notes.
Home Features FAQs About
localhost:5000/dashboard/item/644bc56fcf0fb50aacc22c4e

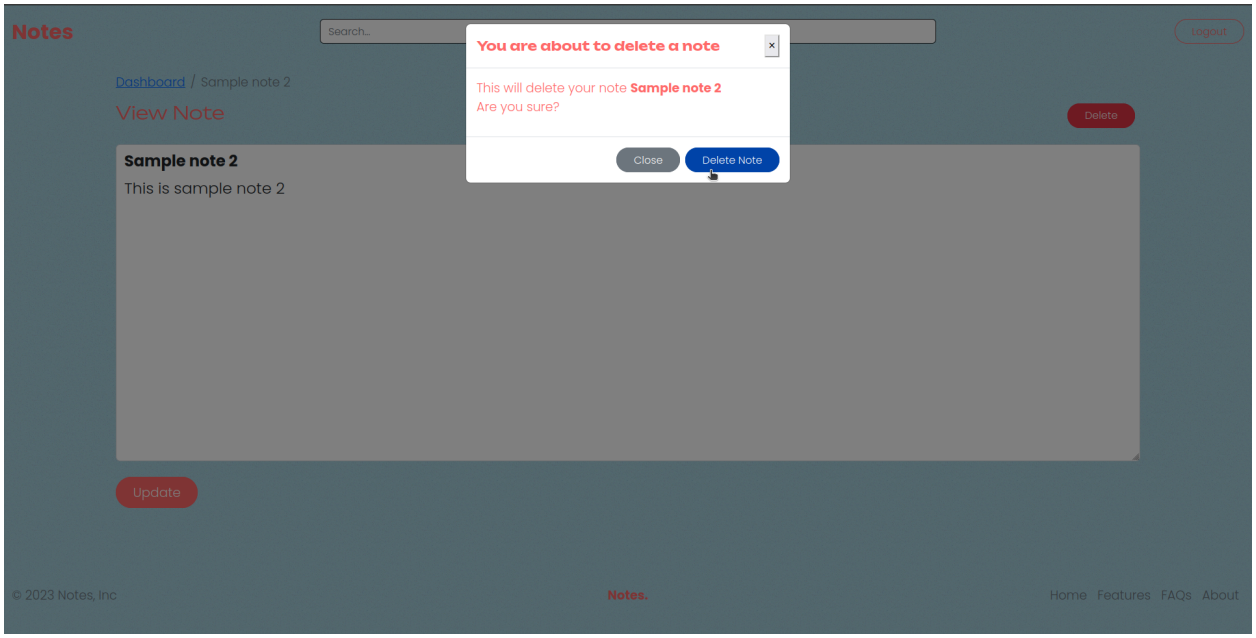

Notes
Search..
Logout
Dashboard / Sample Note
View Note
Delete
Sample Updated Note
This is my first sample note //updated
Update
© 2023 Notes, Inc
Notes.
Home Features FAQs About

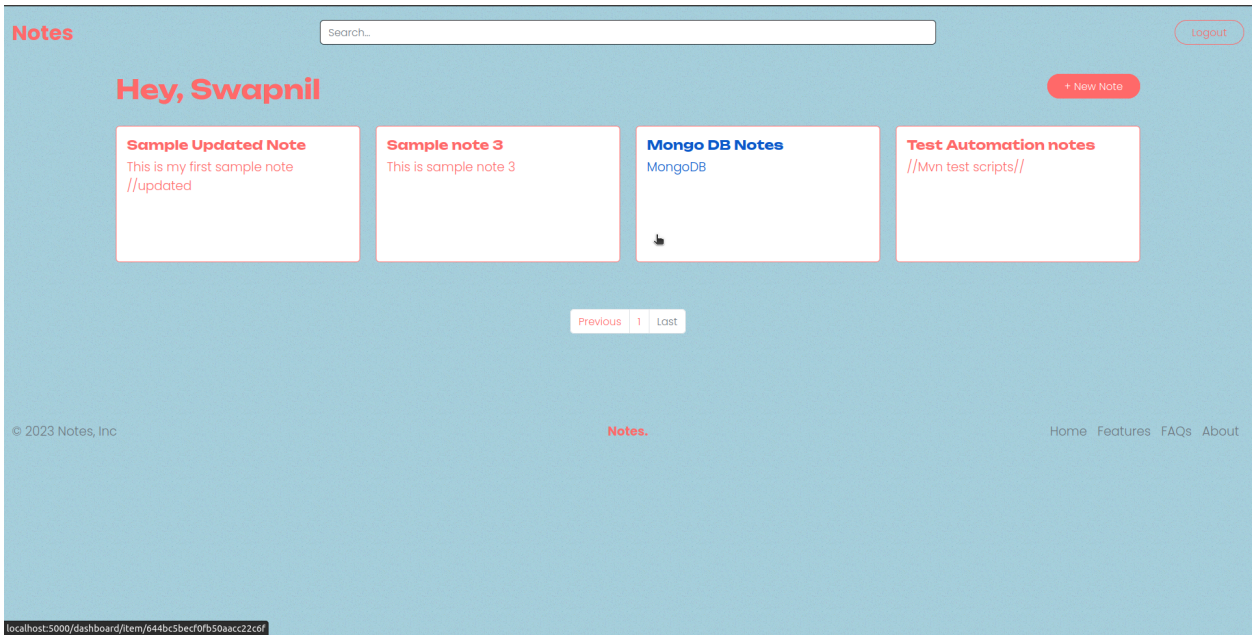


Deleting note

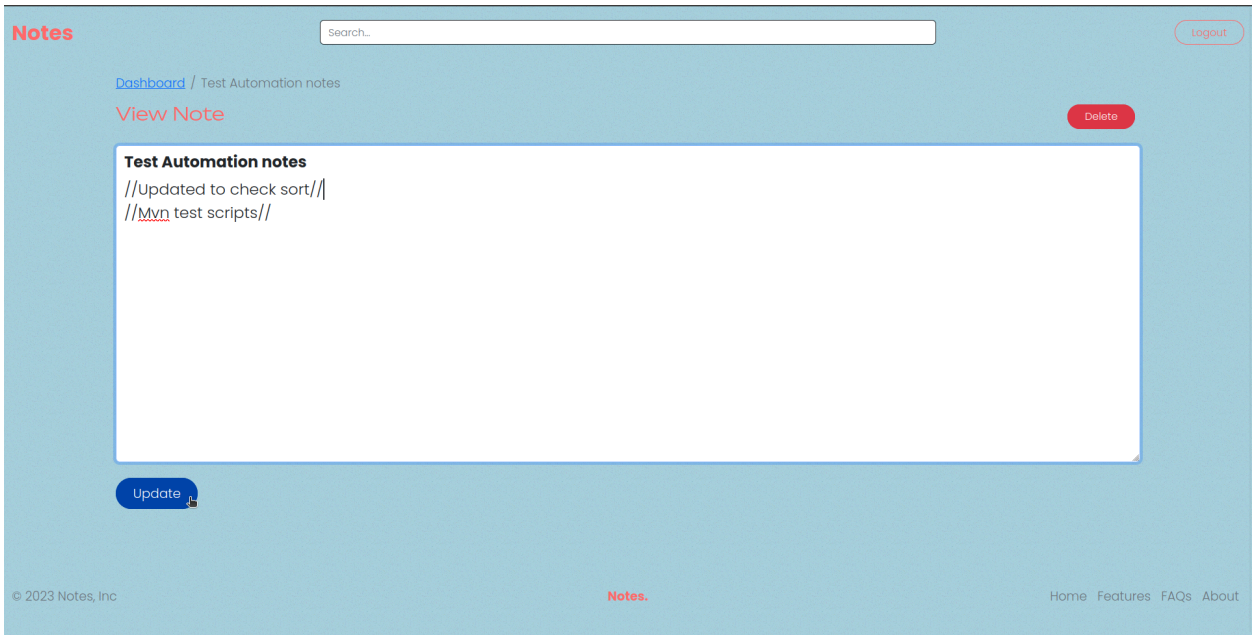


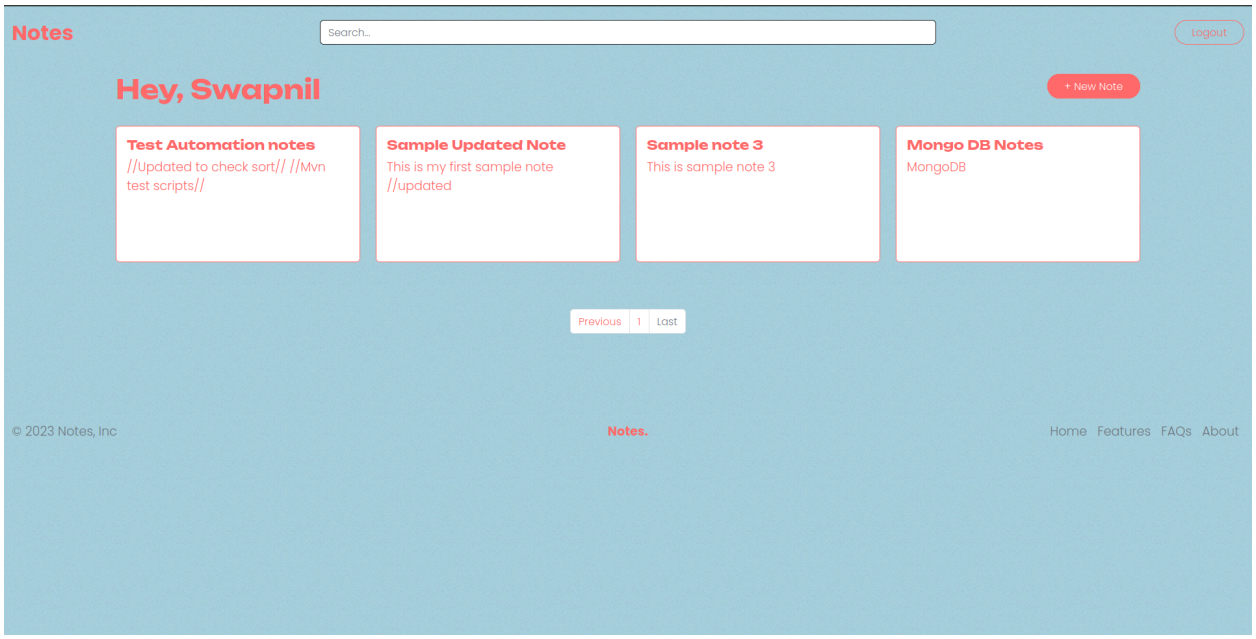

Notes
Search...
Logout
Dashboard / Sample note 2
View Note
Delete
Sample note 2
This is sample note 2
Update
© 2023 Notes, Inc
Notes.
Home Features FAQs About


Notes
Search...
Logout
Dashboard / Sample note 2
View Note
Delete
Sample note 2
This is sample note 2
Update
You are about to delete a note
This will delete your note Sample note 2
Are you sure?
Close
Delete Note
© 2023 Notes, Inc
Notes.
Home Features FAQs About

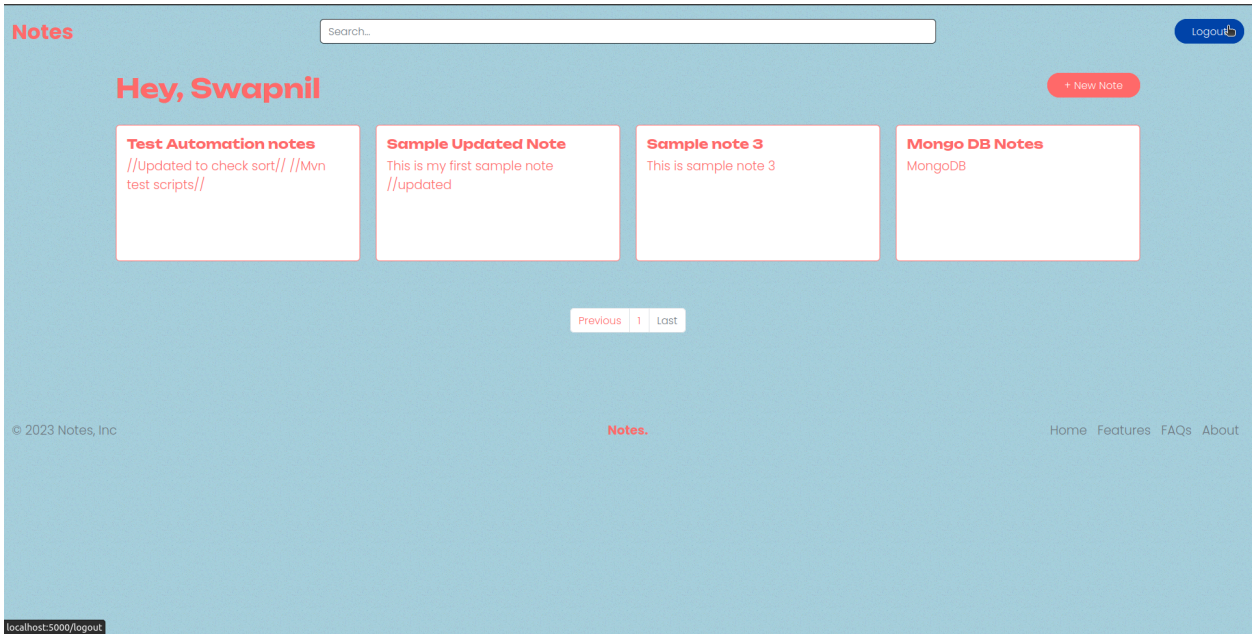


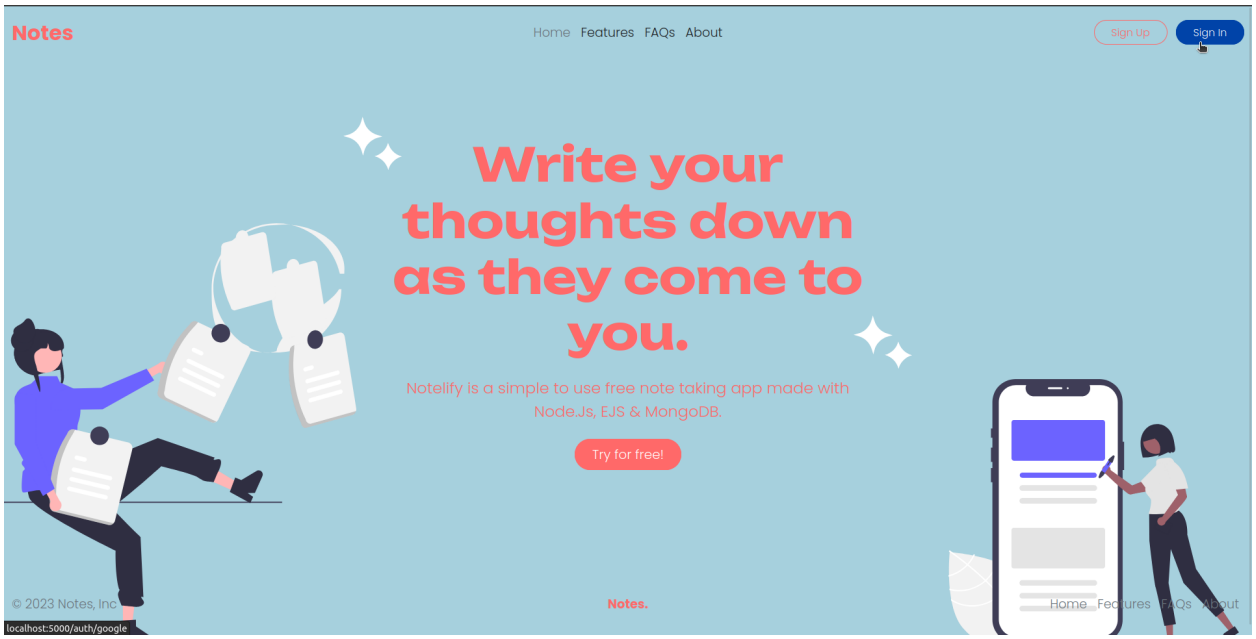
Sorting (Latest update note comes at top)

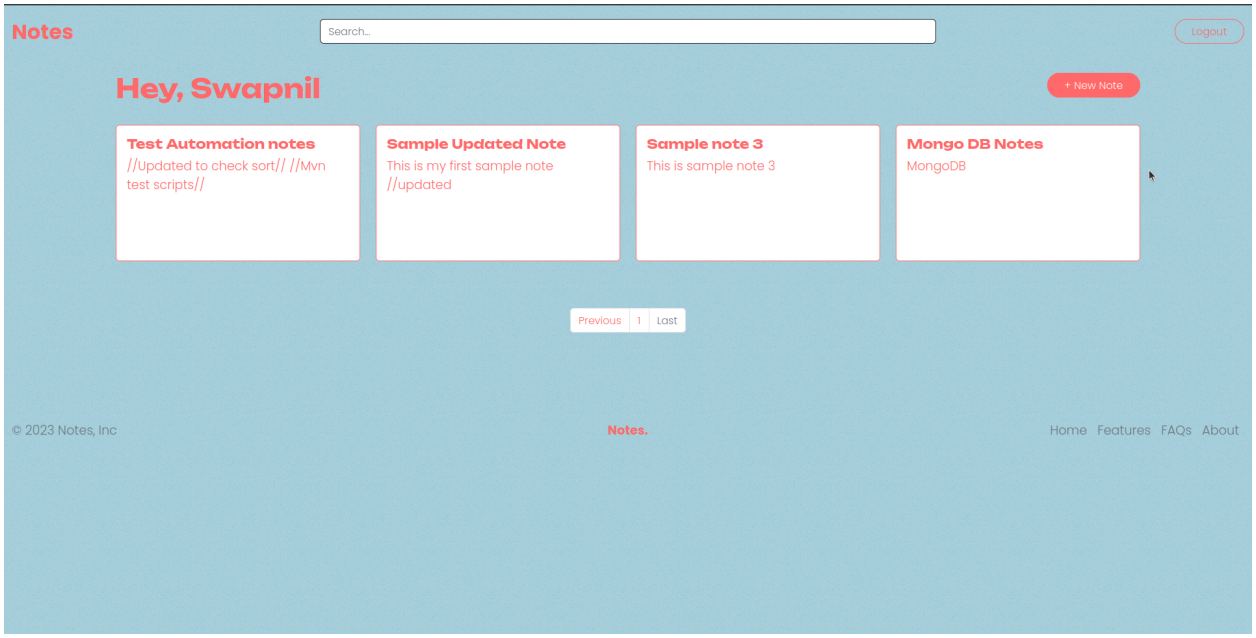




Relogin and check for saved notes

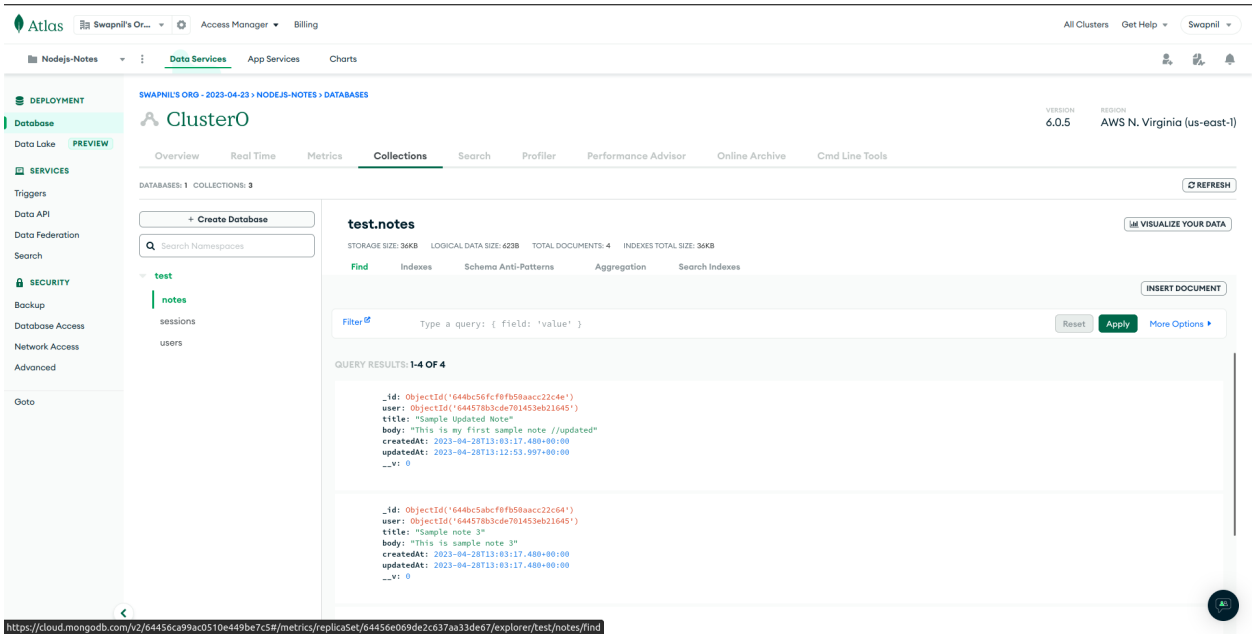






MongoDB Collections

//Overview//



//Notes//


```
_id: ObjectId('644bc56fcf0fb50aacc22c4e')
user: ObjectId('644578b3cde701453eb21645')
title: "Sample Updated Note"
body: "This is my first sample note //updated"
createdAt: 2023-04-28T13:03:17.480+00:00
updatedAt: 2023-04-28T13:12:53.997+00:00
__v: 0
```

```
_id: ObjectId('644bc5abcf0fb50aacc22c64')
user: ObjectId('644578b3cde701453eb21645')
title: "Sample note 3"
body: "This is sample note 3"
createdAt: 2023-04-28T13:03:17.480+00:00
updatedAt: 2023-04-28T13:03:17.480+00:00
__v: 0
```

```
_id: ObjectId('644bc5becf0fb50aacc22c6f')
user: ObjectId('644578b3cde701453eb21645')
title: "Mongo DB Notes"
body: "MongoDB"
createdAt: 2023-04-28T13:03:17.480+00:00
```


//Sessions//

```
_id: "BMtGbufAetikrIJtPSqL3bcDggcSYyFt"
expires: 2023-05-07T18:22:56.832+00:00
session: "{"cookie":{"originalMaxAge":null,"expires":null,"httpOnly":true,"path"…"

_id: "RGiK1YLOoRWJyMKIryTQoP9wStwfKgNH"
expires: 2023-05-07T19:32:41.202+00:00
session: "{"cookie":{"originalMaxAge":null,"expires":null,"httpOnly":true,"path"…"

_id: "jPrpKcKL5-Ze03Da57xDBTbDVIUM9-eo"
expires: 2023-05-07T21:28:51.745+00:00
session: "{"cookie":{"originalMaxAge":null,"expires":null,"httpOnly":true,"path"…"

_id: "NQEAyYq0IBV4RRJzf7Ck6CPzcw_ZtCpz"
expires: 2023-05-07T21:44:45.779+00:00
session: "{"cookie":{"originalMaxAge":null,"expires":null,"httpOnly":true,"path"…"
```


//Users//

QUERY RESULTS: **1-1 OF 1**


```
_id: ObjectId('644578b3cde701453eb21645')
googleId: "115321965624472479343"
displayName: "Swapnil Swapnil"
firstName: "Swapnil"
lastName: "Swapnil"
profileImage: "https://lh3.googleusercontent.com/a/AGNmyxYwNBXrFGLl_Hcd79OSfv4b1fFpHB…"
createdAt: 2023-04-23T18:28:03.961+00:00
__v: 0
```