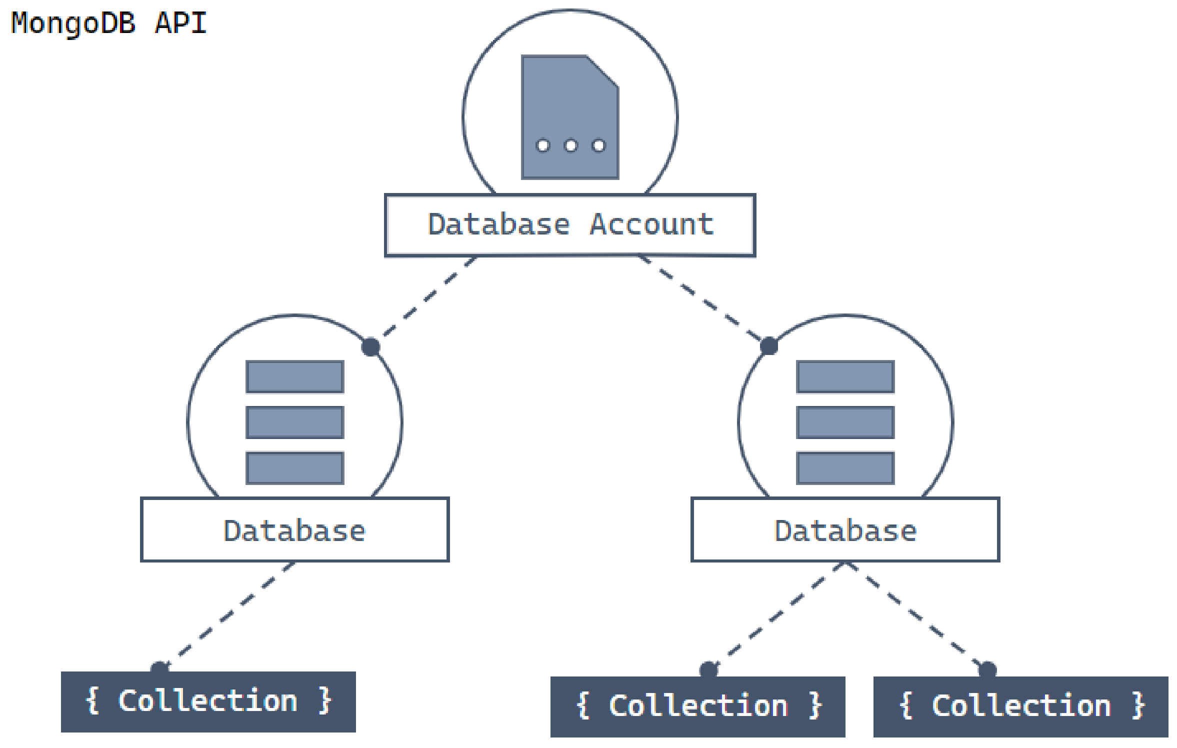
My TODO List website using Azure Cosmos DB for MongoDB

Introduction to Cosmos DB:

Azure Cosmos DB for MongoDB makes it easy to use Azure Cosmos DB as if it were a MongoDB database. Here we can use our existing MongoDB skills and continue to use our favorite MongoDB drivers, SDKs, and tools by pointing our application to the connection string for our account using the API for MongoDB.

Hierarchy of resources in Cosmos DB:



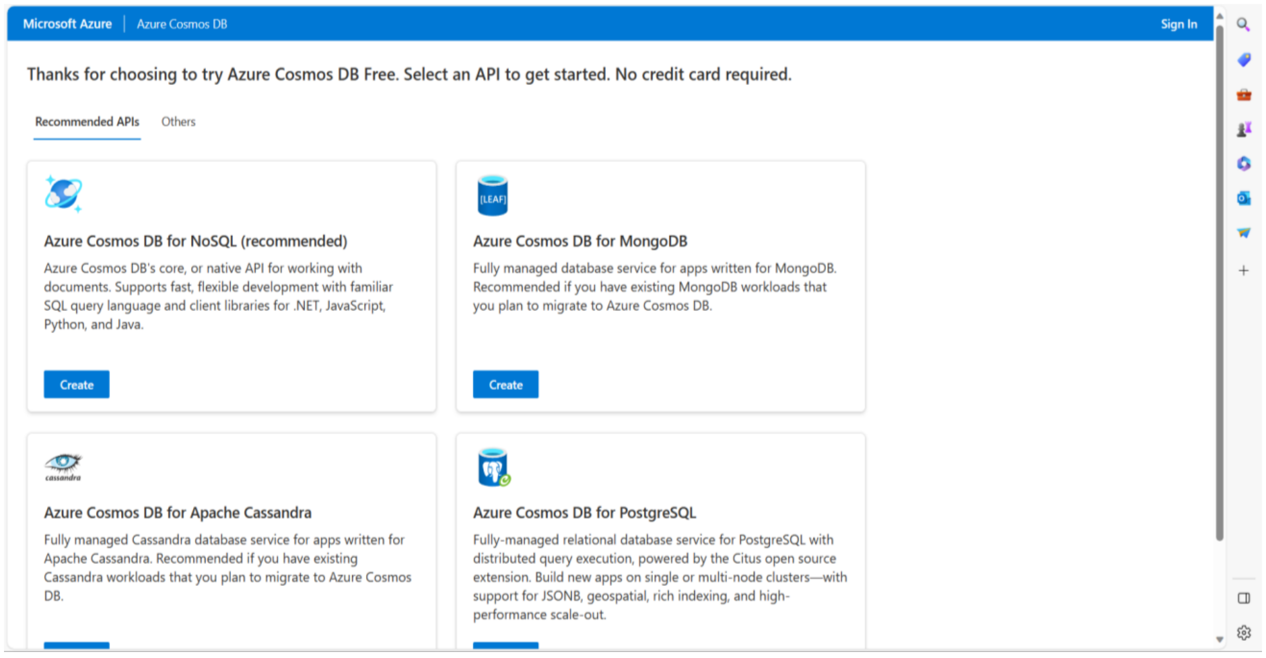
In order to interact with these resources there are three different Mongo DB classes:

- Mongo Client is the class that provides a client-side logical representation for the API for MongoDB layer on Azure Cosmos DB.

- DB is the class that is reference to a database that may, or may not exist in the server yet.
- The collection is validated server-side when we attempt to work with it.

Cosmos DB account Creation:

Step1 : Open Azure Cosmos DB. Click on "Try Azure Cosmos DB for free" .You will be directed to the page where you can create your account and launch a quick start.



Thanks for choosing to try Azure Cosmos DB Free. Select an API to get started. No credit card required.

Azure Cosmos DB for NoSQL (recommended)

Azure Cosmos DB's core, or native API for working with documents. Supports fast, flexible development with familiar SQL query language and client libraries for .NET, JavaScript, Python, and Java.

Azure Cosmos DB for MongoDB

Fully managed database service for apps written for MongoDB. Recommended if you have existing MongoDB workloads that you plan to migrate to Azure Cosmos DB.

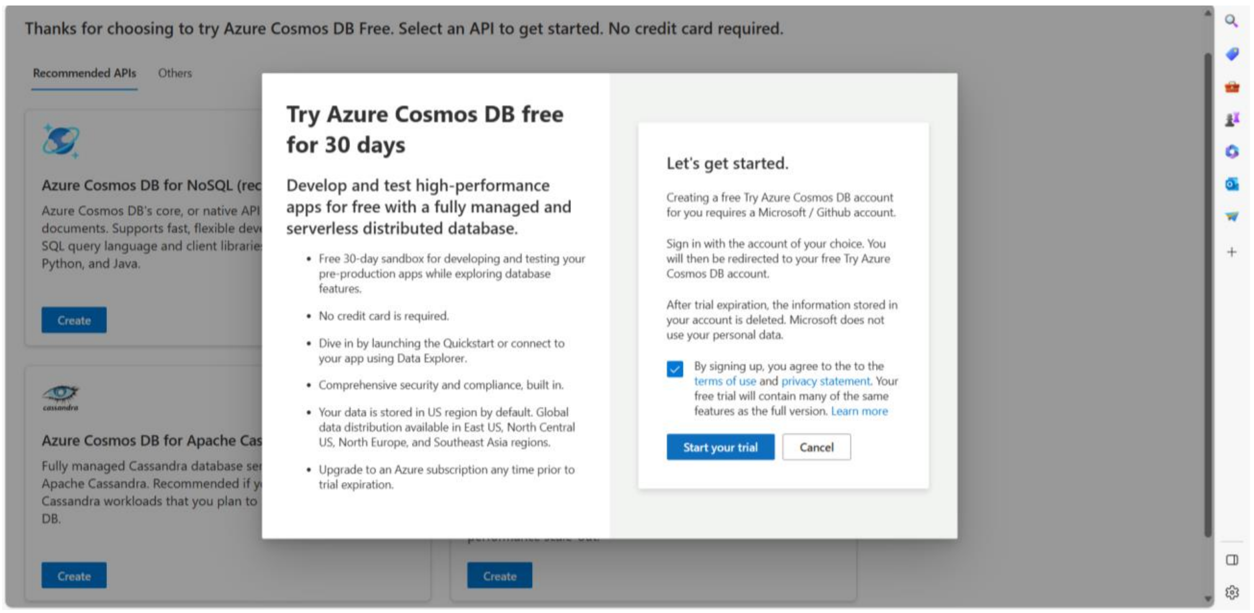
Azure Cosmos DB for Apache Cassandra

Fully managed Cassandra database service for apps written for Apache Cassandra. Recommended if you have existing Cassandra workloads that you plan to migrate to Azure Cosmos DB.

Azure Cosmos DB for PostgreSQL

Fully-managed relational database service for PostgreSQL with distributed query execution, powered by the Citus open source extension. Build new apps on single or multi-node clusters—with support for JSONB, geospatial, rich indexing, and high-performance scale-out.

Step 2: You can start your first trail to create the new Collections.


Thanks for choosing to try Azure Cosmos DB Free. Select an API to get started. No credit card required.

Recommended APIs Others

Azure Cosmos DB for NoSQL (rec

Azure Cosmos DB's core, or native API documents. Supports fast, flexible dev SQL query language and client librarie Python, and Java.

Create

Azure Cosmos DB for Apache Cas

Fully managed Cassandra database se Apache Cassandra. Recommended if y Cassandra workloads that you plan to DB.

Create

Create

Try Azure Cosmos DB free for 30 days

Develop and test high-performance apps for free with a fully managed and serverless distributed database.

- Free 30-day sandbox for developing and testing your pre-production apps while exploring database features.
- No credit card is required.
- Dive in by launching the Quickstart or connect to your app using Data Explorer.
- Comprehensive security and compliance, built in.
- Your data is stored in US region by default. Global data distribution available in East US, North Central US, North Europe, and Southeast Asia regions.
- Upgrade to an Azure subscription any time prior to trial expiration.

Let's get started.

Creating a free Try Azure Cosmos DB account for you requires a Microsoft / Github account.

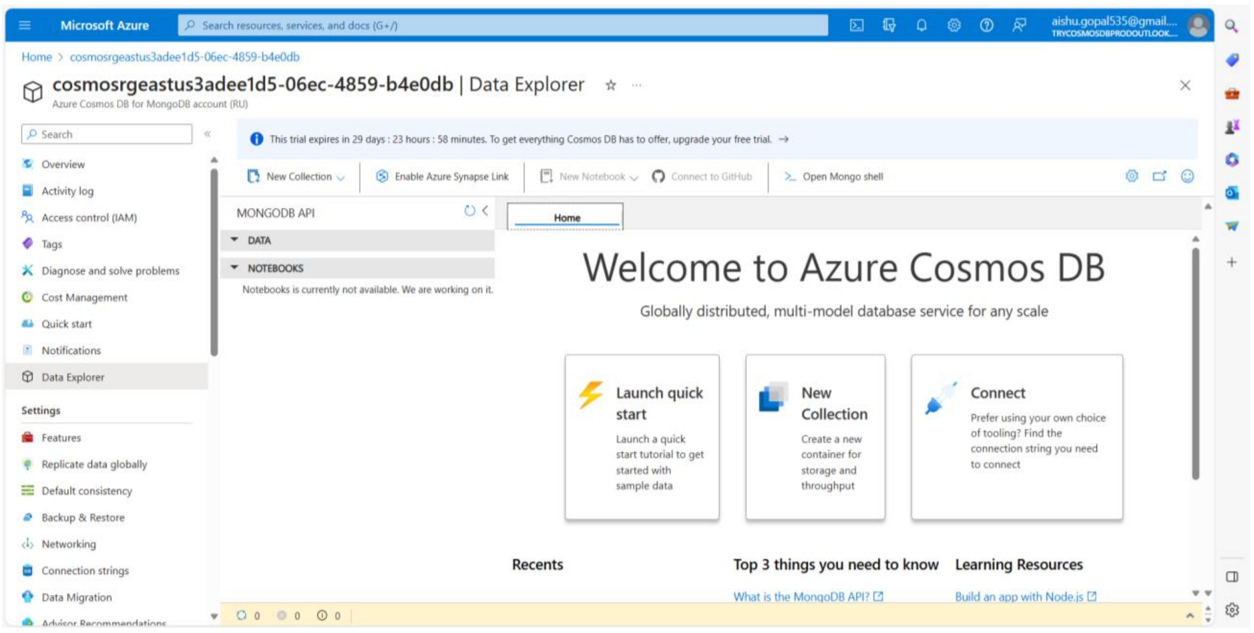
Sign in with the account of your choice. You will then be redirected to your free Try Azure Cosmos DB account.

After trial expiration, the information stored in your account is deleted. Microsoft does not use your personal data.

[✓] By signing up, you agree to the to the terms of use and privacy statement. Your free trial will contain many of the same features as the full version. Learn more

Start your trial Cancel

Step 3: Open the Data Explorer pane. Select "New Container". Indicate whether you are creating a new database or using an existing one. Enter a container ID.


Microsoft Azure Search resources, services, and docs (G+/) aishu.gopal535@gmail.... TRYCOSMOSDBPRODOUTLOOK...

Home > cosmosrgeastus3adee1d5-06ec-4859-b4e0db

cosmosrgeastus3adee1d5-06ec-4859-b4e0db | Data Explorer

Azure Cosmos DB for MongoDB account (RU)

This trial expires in 29 days : 23 hours : 58 minutes. To get everything Cosmos DB has to offer, upgrade your free trial. →

Notebooks is currently not available. We are working on it.

Welcome to Azure Cosmos DB

Globally distributed, multi-model database service for any scale

Launch quick start Launch a quick start tutorial to get started with sample data

New Collection Create a new container for storage and throughput

Connect Prefer using your own choice of tooling? Find the connection string you need to connect

Recents

Top 3 things you need to know

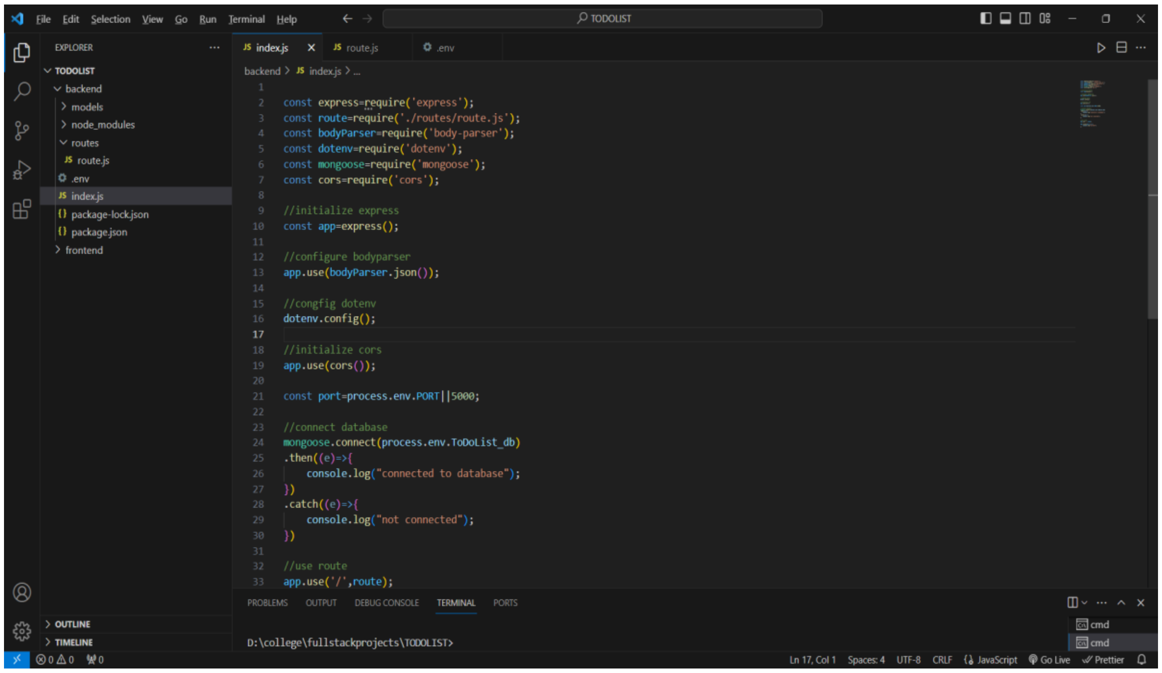
What is the MongoDB API?

Learning Resources

Build an app with Node.js

Connecting the project with Cosmos DB:

- In order to connect Cosmos DB with your project follow the below steps:
  - ✓ Open your backend part of your project.
  - ✓ Install the MongoDB driver into your project. For example, if you are using Node.js, you can install the driver by running the following command in your terminal: npm install mongoose.
  - ✓ Import the MongoDB libraries into your project. The exact steps for doing this will depend on your programming language and development environment. For example, if you are using Node.js, you can import the MongoDB driver by adding the following line to your code: const mongoose= require('mongoose').



```
const express=require('express');
const route=require('./routes/route.js');
const bodyParser=require('body-parser');
const dotenv=require('dotenv');
const mongoose=require('mongoose');
const cors=require('cors');

//initialize express
const app=express();

//configure bodyparser
app.use(bodyParser.json());

//congfig dotenv
dotenv.config();

//initialize cors
app.use(cors());

const port=process.env.PORT||5000;

//connect database
mongoose.connect(process.env.ToDoList_db)
.then((e)=>{
    console.log("connected to database");
})
.catch((e)=>{
    console.log("not connected");
})

//use route
app.use('/',route);
```

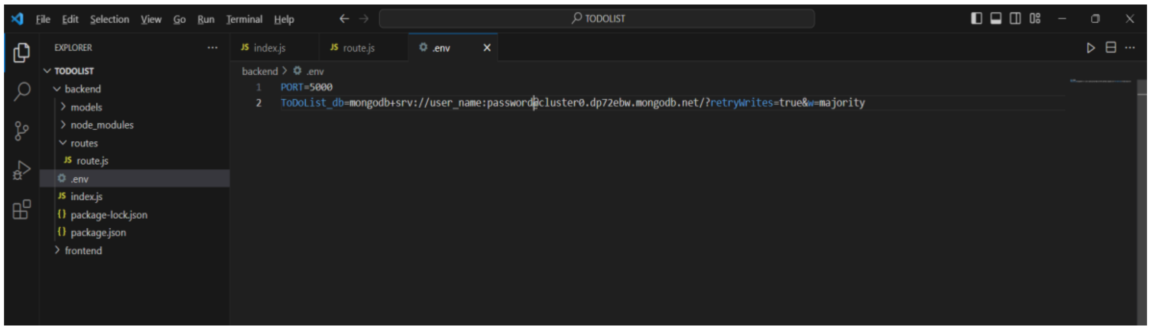
```
D:\college\fullstackprojects\TODOLIST>
```

- To create a new collection in Cosmos DB for your project, you can follow these steps:
  - ✓ To create a new collection in Cosmos DB for your project, you can follow these steps:
  - ✓ Open Azure Cosmos DB.
  - ✓ Create a new Azure Cosmos DB account or select an existing one.
  - ✓ Open the Data Explorer pane.
  - ✓ Select “New Container”.

    - ✓ Indicate whether you are creating a new database or using an existing one.
    - ✓ Enter a container ID.

- ➢ To get the URL of a database in Cosmos DB, you can follow these steps:
    - ✓ Open the Data Explorer pane.
    - ✓ Select the database you want to get the URL for.
    - ✓ Click on the “Overview” tab.
    - ✓ The URL of the database will be displayed in the “URI” field and copy it.

- ➢ Open the .env file in your backend project. Locate the line where you want to paste the URL.



```
PORT=5000
ToDoList_db=mongodb+srv://user_name:password@cluster0.dp72ebw.mongodb.net/?retryWrites=true&w=majority
```

- ➢ Replace username and password with your details.

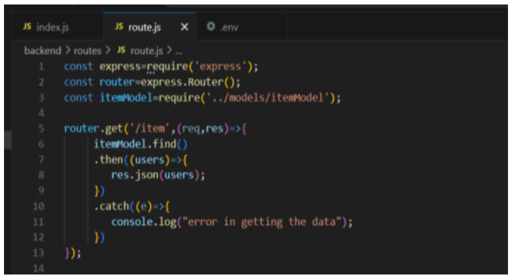
- ➢ This helps us to connect Cosmos DB with the project to access the database

Moving towards Todo list Project:

A To-do list is a collection of tasks designed for day-to-day organization. This project is built using ReactJS and NodeJS, with Azure Cosmos DB serving as the MongoDB database.

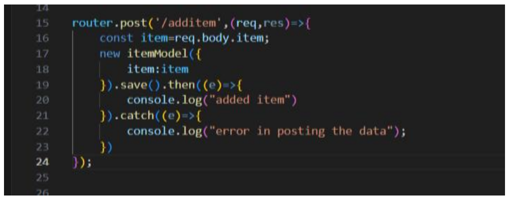
- ➢ ReactJS is employed for the complete front-end implementation, providing the user interface, while NodeJS adds the necessary backend functionality.
- ➢ The project encompasses various processes, including adding new tasks, retrieving existing data, updating task details, and deleting tasks from the database.
- ➢ API commands are utilized to interact with the Cosmos DB database, facilitating seamless communication between the application and the data store.

- For retrieving data from a Cosmos DB collection, the project utilizes the GET method. The syntax for the get() method is exemplified as follows:



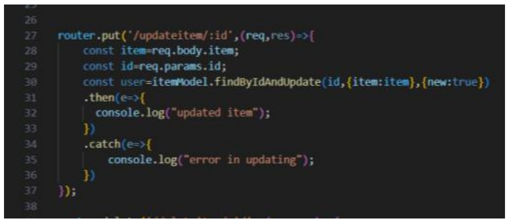
```javascript
const express=require('express');
const router=express.Router();
const itemModel=require('../models/itemModel');

router.get('/item',(req,res)=>{
    itemModel.find()
    .then((users)=>{
        res.json(users);
    })
    .catch((e)=>{
        console.log("error in getting the data");
    })
});

```

- To insert new data into a Cosmos DB database, you can use the **POST** method. The syntax for the post() method is as follows:



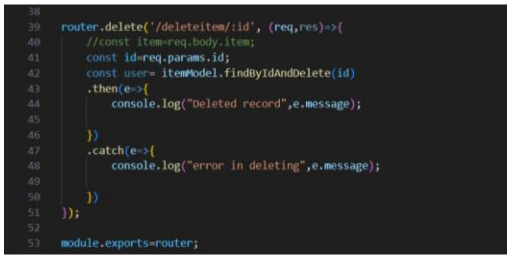
```javascript
router.post('/additem',(req,res)=>{
    const item=req.body.item;
    new itemModel({
        item:item
    }).save().then((e)=>{
        console.log("added item")
    }).catch((e)=>{
        console.log("error in posting the data");
    })
});
```

- To update or modify existing data in a Cosmos DB collection, you can use the **PUT** Method The syntax for the put() method is as follows:



```javascript
router.put('/updateitem/:id',(req,res)=>{
    const item=req.body.item;
    const id=req.params.id;
    const user=itemModel.findByIdAndUpdate(id,{item:item},{new:true})
    .then(e=>{
      console.log("updated item");
    })
    .catch(e=>{
        console.log("error in updating");
    })
});
```

- To delete documents from a Cosmos DB collection, you can use the **DELETE** Method. The syntax for the delete() method is as follows:



```javascript
router.delete('/deleteitem/:id', (req,res)=>{
    //const item=req.body.item;
    const id=req.params.id;
    const user= itemModel.findByIdAndDelete(id)
    .then(e=>{
        console.log("Deleted record",e.message);

    })
    .catch(e=>{
        console.log("error in deleting",e.message);

    })
});

module.exports=router;
```

Glimpses of TODO list Project:

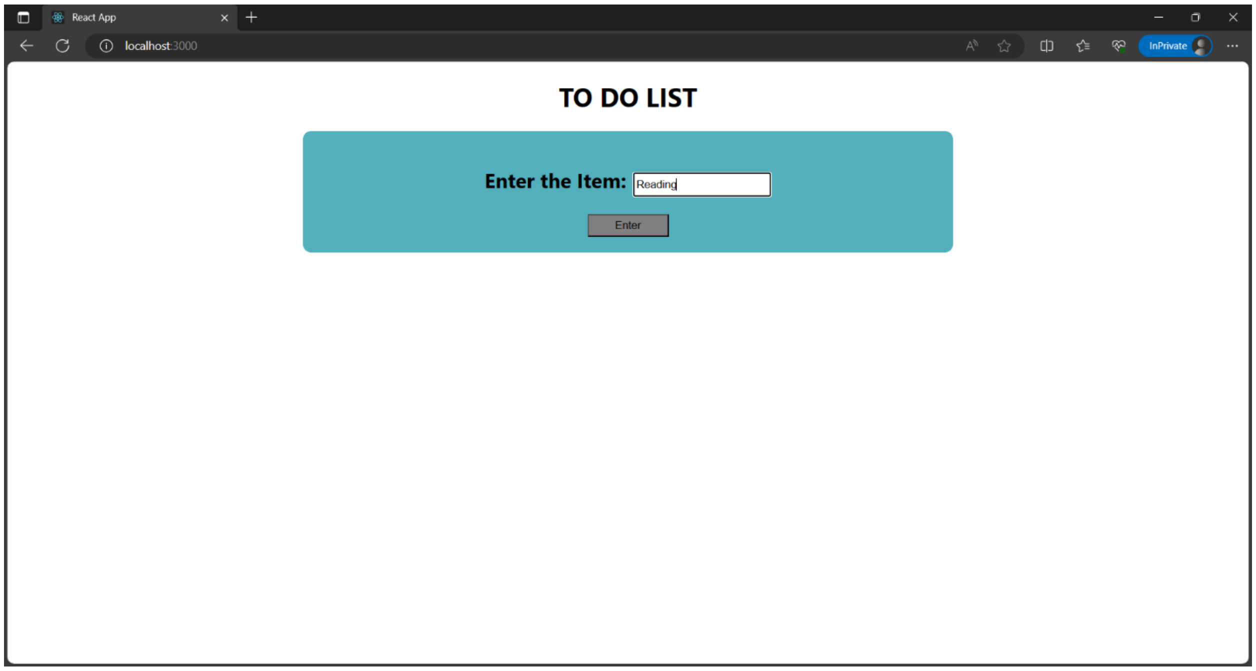


Figure 1:Front page of Todo list Website

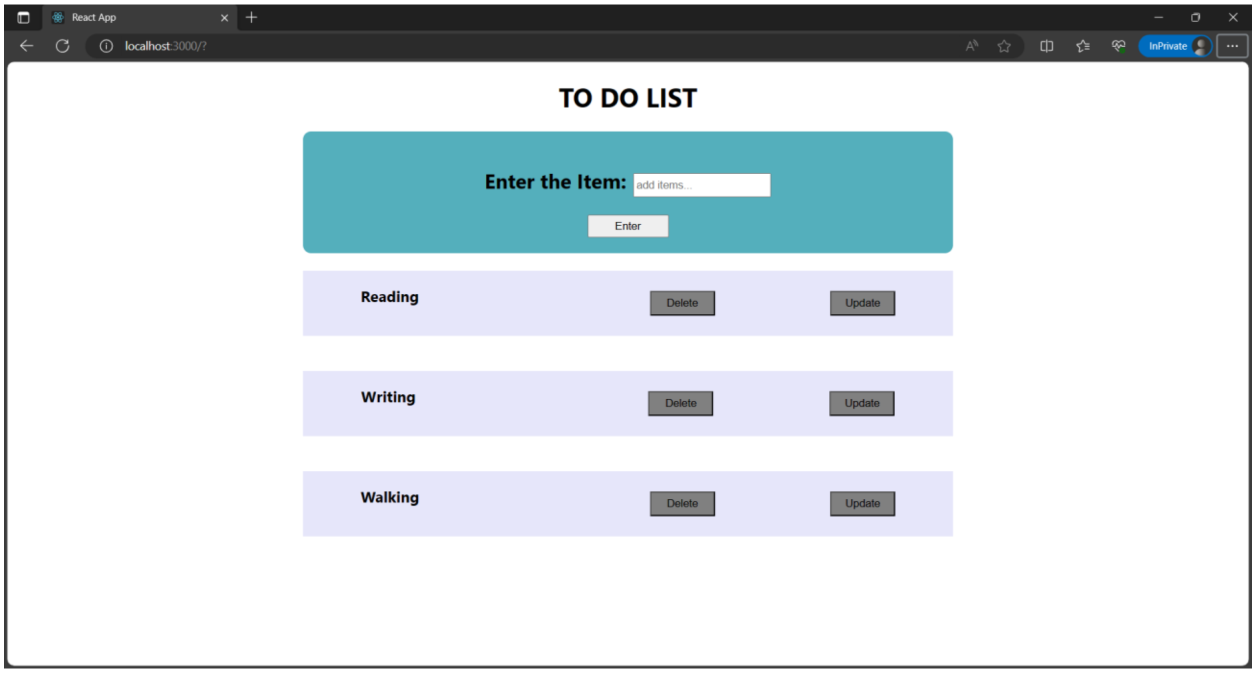


Figure 2:Added items to the Todo list

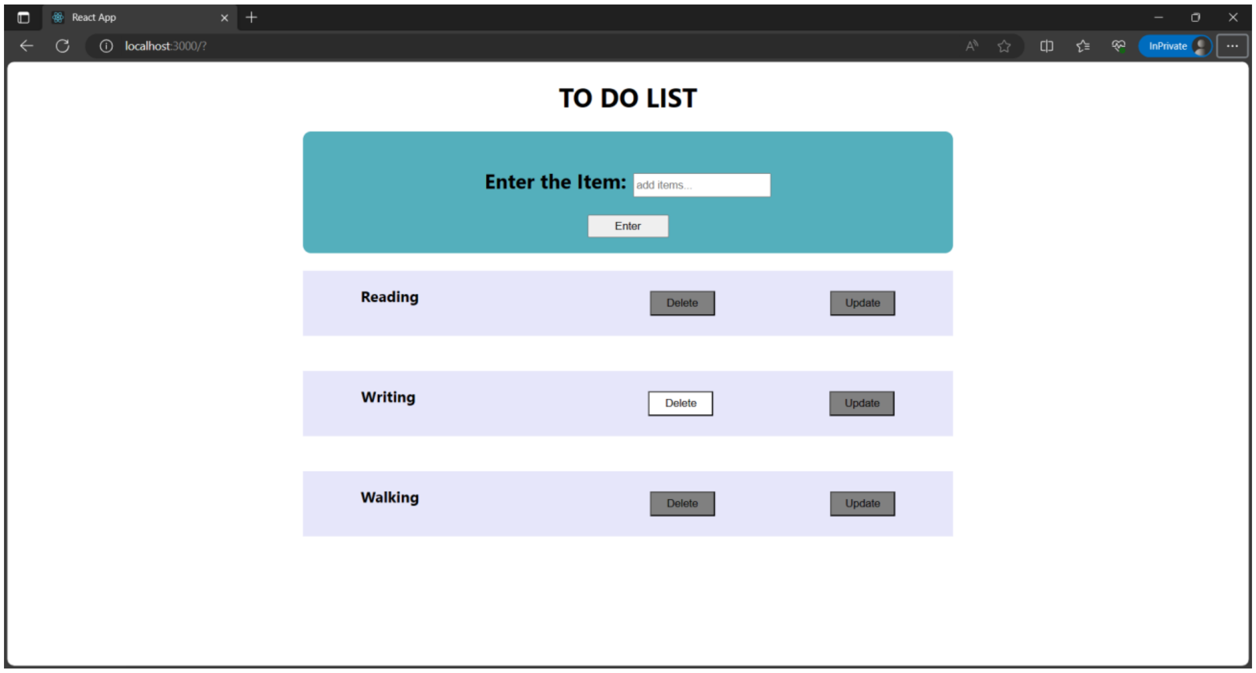


Figure 3:Deleting the items in the List

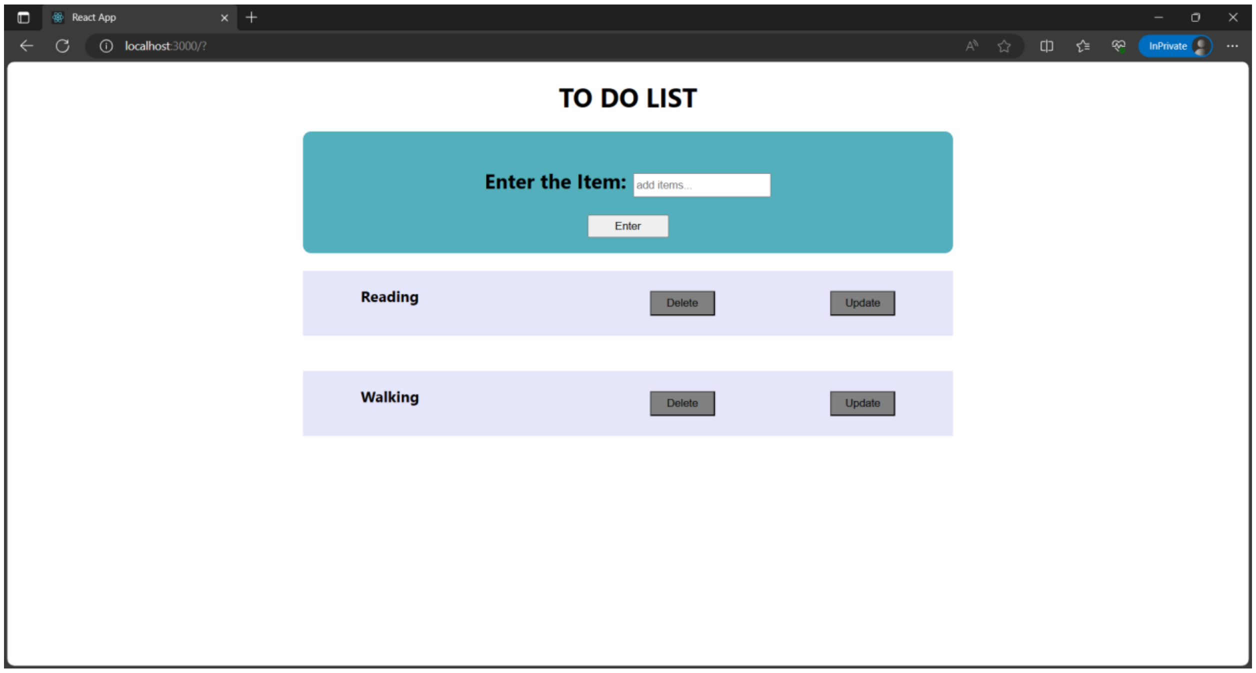


Figure 4:Data has been removed from the Database

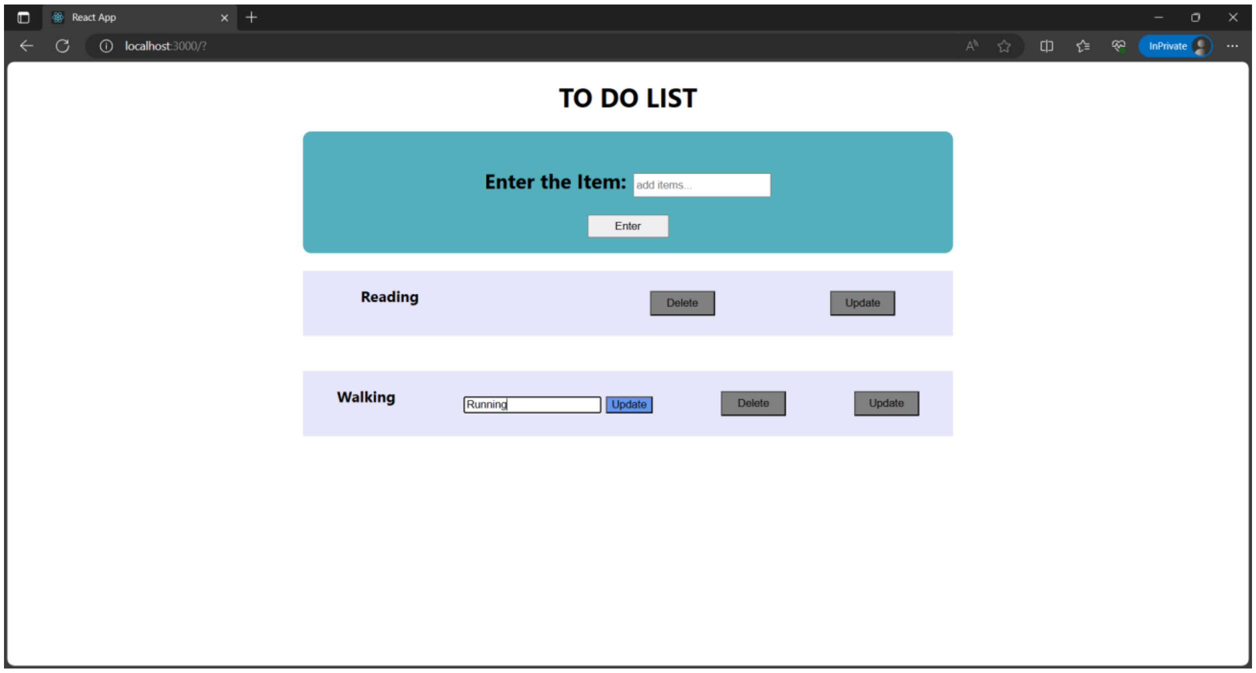


Figure 5:Updating the items in the List

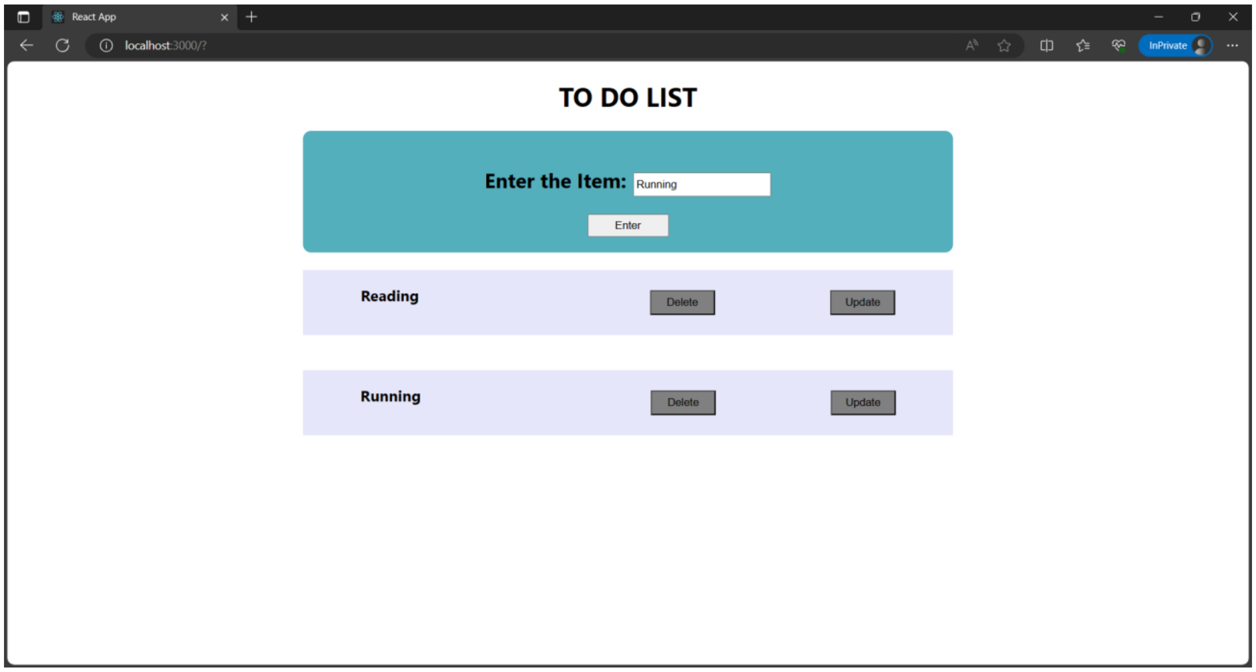


Figure 6:Updated items in the database

This Project code can be accessed from the link below:

GitHub Repo Link

To clone the repository, please follow these steps:

1. Open the GitHub repository link provided.
2. Click on the green "Code" button located on the right side of the page.
3. Copy the URL of the repository.
4. Open your terminal or Git Bash.
5. Navigate to the directory where you want to store the cloned repository.
6. Type the following command and press Enter:

   git clone https://github.com/AishwaryaReddy7547/TODO-List.git

Make sure Git is installed on your machine before executing the above command. If Git is not installed, you can download and install it from the official Git website: Git Downloads.