Amazon Clone App

1. ***Created separate file for route –***

Routers.dart

Route<dynamic> generateRoute(RouteSettings routeSettings) {

  switch (routeSettings.name) {

    case AuthScreen.routeName:

      return MaterialPageRoute(

        settings: routeSettings,

        builder: (\_) => const AuthScreen(),

      );

    default:

      return MaterialPageRoute(

        builder: (\_) => const Scaffold(

          body: Center(

            child: Text("Page Not Found!"),

          ),

        ),

      );

  }

}

Main.dart

void main() {

  runApp(const MyApp());

}

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

//code

       onGenerateRoute: (settings) => generateRoute(settings),

      home: //Widget,

    );

  }

}

1. **Downloading Node js –**

**Link to download -** [**https://nodejs.org/en/download/prebuilt-installer/current**](https://nodejs.org/en/download/prebuilt-installer/current)

**Check in command prompt -**

**C:\Users\Asus>node --version**

**v20.12.2**

1. **To use node.js in our project-**

**Create folder “Server” open in in terminal and follow steps-**

…. /Amazon\_Clone/amazon/server (main)

$ npm init

This utility will walk you through creating a package.json file.

It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields

and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and

save it as a dependency in the package.json file.

Press ^C at any time to quit.

package name: (server)

version: (1.0.0)

description:

entry point: (index.js)

test command:

git repository:

keywords:

author:

license: (ISC)

About to write to D:\IMPMyWorkplace\GitHub\Amazon\_Clone\amazon\server\package.json:

{

"name": "server",

"version": "1.0.0",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC",

"description": ""

}

Is this OK? (yes)

**After this one file will create in server package.json**

**Create a file index.js and run it:**

Index.js

console.log("hi iam json");

….. /Amazon\_Clone/amazon/server (main)

$ node index.js

hi iam json

1. **Now download dependencies http, express, mangoose –**

….. /Amazon\_Clone/amazon/server (main)

$ npm i http express mongoose

**added 85 packages, and audited 86 packages in 17s**

**after this automatically dependencies are added**

package.json

//code

"dependencies": {

    "express": "^4.19.2",

    "http": "^0.0.1-security",

    "mongoose": "^8.4.1"

  }

1. **Create Api**

Index.js

const express=require("express");

const PORT=3000;

const app=express();//creating instance of express (aboue)

//creating API

// app.listen(PORT,"0.0.0.0",()=>{console.log('connected at port ${PORT}')}); use back tick instead

app.listen(PORT,"0.0.0.0",()=>{console.log(`connected at port ${PORT}`);

});

To run-

…/Amazon\_Clone/amazon/server (main)

$ node index.js

connected at port 3000

1. **Download nodemon package as dev for better run of our api –**

**To add dev packages command is different**

…/Amazon\_Clone/amazon/server (main)

$ npm i nodemon --save-dev

added 29 packages, and audited 115 packages in 7s

**after this one change in package.json-**

"devDependencies": {

    "nodemon": "^3.1.2"

  }

**Now to run index.js fire command npm run dev ->**

**For better run add script first at “dev” and for better deployment add at “start”**

Package.json

 "scripts": {

    "dev": "nodemon ./index.js",

    "start": "node ./index.js"

  },

**To run 🡪**

/Amazon\_Clone/amazon/server (main)

$ npm run dev

npm error Missing script: "dev"

npm error

…/Amazon\_Clone/amazon/server (main)

$ npm run dev

> server@1.0.0 dev

> nodemon ./index.js

[nodemon] 3.1.2

[nodemon] to restart at any time, enter `rs`

[nodemon] watching path(s): \*.\*

[nodemon] watching extensions: js,mjs,cjs,json

[nodemon] starting `node ./index.js`

connected at port 3000

1. **Create a get request:**

**IMP: remove port “0:0:0:0”**

**Package.json**

//get request

app.get('/hello', (req, res) => {

    res.send({name:"Rowdy"},);

});

To run : <http://localhost:3000/hello> in chrome

to send json data:

app.get('/json', (req, res) => {

    res.json({ name: "Rowdy" },);

});

1. **Creating routers in different files and importing it in main js**

**Ex:**

Routers/auth.js

const express=require("express");

const authRouter=express.Router();

authRouter.get('/user',(req,res)=>{

res.json({msg:"Suraj"});

});

module.exports=authRouter;

index.js

//imports from packages

const express=require("express");

const { get } = require("http");

//imports from other files

const authRouter=require("./routers/auth");

//initalization

const PORT=3000;

const app=express();

//middleware

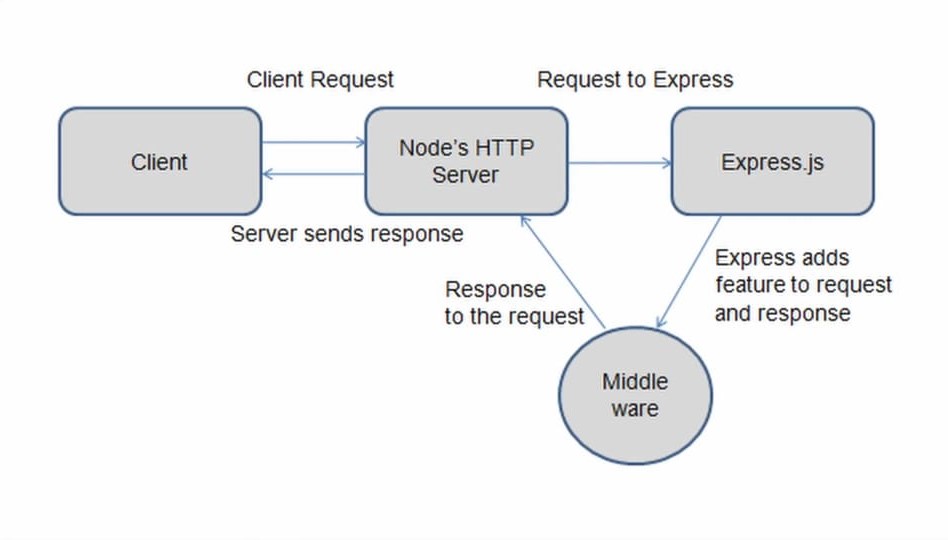
app.use(authRouter);

// app.listen(PORT,"0.0.0.0",()=>{console.log('connected at port ${PORT}')}); use back tick instead

app.listen(PORT,()=>{console.log(`connected at port ${PORT}`);

});

1. **What is middleware**



1. **To create post request-**

auth.js

const express=require("express");

const authRouter=express.Router();

authRouter.post('/api/users/',(req,res)=>{

//get data from user

const {name,email,password} =req.body;

});

module.exports=authRouter;

1. **Create a mangodb account and then create project which is simple steps and After creating go to security and do not forgot to user it for “mongo db native drivers” and port “0.0.0.0” witch means access from anywhere**

**Now go to database then pr name then connect then copy Add your connection string into your application code and copy that-**

“mongodb+srv://suraj:<password>@flutteramazoncloneapp.zspb2e1.mongodb.net/?retryWrites=true&w=majority&appName=FlutterAmazonCloneApp”

In place of <password> add suraj123 that was added while creating project

1. **Creating model and sign up for user**

models/user\_model.js

const mongoose = require('mongoose');

//creating scema

const userScema = mongoose.Schema({

    name: {

        required: true,

        type: String,

        trim: true,

    },

    email: {

        required: true,

        type: String,

        trim: true,

        validate: {

            validator: (value) => {

                const re =

                    /^(([^<>()[\]\.,;:\s@\"]+(\.[^<>()[\]\.,;:\s@\"]+)\*)|(\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^<>()[\]\.,;:\s@\"]{2,})$/i;

                return value.match(re);

            },

            message: "Please enter valid email",

        },

    },

    password: {

        required: true,

        type: String,

    validate: {

    validator: (value) => {

        const pass =

            /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*(\W|\_)).{6,}$/;

        return value.match(pass);

    },

        message: "Please enter strong password",

        },

    },

    address: {

        type: String,

        default: '',

    },

    type: {

        type: String,

        default: 'user',

    },

    //cart

});

//creating model

const User = mongoose.model("User", userScema);

module.exports=User;

**then for sign up make changes in auth.js**

auth.js

const express = require("express");

const bcryptjs = require("bcryptjs");

const User = require("../models/user\_model");

const authRouter = express.Router();

const jwt = require("jsonwebtoken");

//sign up

authRouter.post("/api/signup", async (req, res) => {

    //get data from user

    try {

        const { name, email, password } = req.body;

        const existingUser = await User.findOne({ email });

        if (existingUser) {

            return res

                .status(400)

                .json({ msg: "user with same email already exists!" });

        }

        //hash password

        const hashedPassword = await bcryptjs.hash(password, 8);

        //add user feilds in model

        let user = new User({

            email,

            name,

            password: hashedPassword,

        });

        //save user and

        //post data to database

        user = await user.save();

        //return the data to user

        res.json(user);

    } catch (e) {

        res.status(500).json({ error: e.message });

    }

});

module.exports = authRouter;

then to use this service in flutter create new file auth\_service.dart also create a json model of user to save all stuff in dart user.dart(db contain \_id do not forgot to change)

user.dart

import 'dart:convert';

class User {

  final String id;

  final String name;

  final String email;

  final String password;

  final String address;

  final String type;

  final String token;

  User({

    required this.id,

    required this.name,

    required this.email,

    required this.password,

    required this.address,

    required this.type,

    required this.token,

  });

  Map<String, dynamic> toMap() {

    return {

      'id': id,

      'name': name,

      'email': email,

      'password': password,

      'address': address,

      'type': type,

      'token': token,

    };

  }

  factory User.fromMap(Map<String, dynamic> map) {

    return User(

      id: map['\_id'] ?? '', //db contain \_id do not forgot to change

      name: map['name'] ?? '',

      email: map['email'] ?? '',

      password: map['password'] ?? '',

      address: map['address'] ?? '',

      type: map['type'] ?? '',

      token: map['token'] ?? '',

    );

  }

  String toJson() => json.encode(toMap());

  factory User.fromJson(String source) => User.fromMap(json.decode(source));

  User copyWith({

    String? id,

    String? name,

    String? email,

    String? password,

    String? address,

    String? type,

    String? token,

  }) {

    return User(

      id: id ?? this.id,

      name: name ?? this.name,

      email: email ?? this.email,

      password: password ?? this.password,

      address: address ?? this.address,

      type: type ?? this.type,

      token: token ?? this.token,

    );

  }

}

auth\_service.dart

import 'package:amazon/models/user.dart';

import 'package:amazon/utils/error\_hanling.dart';

import 'package:amazon/utils/exports.dart';

import 'package:amazon/utils/show\_snakbar.dart';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

class AuthServices {

  //sign up user

  void signUpUser({

    required BuildContext context,

    required String email,

    required String password,

    required String name,

  }) async {

    try {

      User user = User(

          id: '',

          name: name,

          email: email,

          password: password,

          address: '',

          type: '',

          token: '');

      http.Response res = await http.post(

        Uri.parse('$uri/api/signup'),

        body: user.toJson(),

        headers: <String, String>{

          'Content-Type': 'application/json; charset=UTF-8',

        },

      );

      httpErrorHandling(

        response: res,

        context: context,

        onSuccess: () {

          showSnackBar(

              context, "Account is created! login in  with your creditials.");

        },

      );

    } catch (e) {

      showSnackBar(context, e.toString());

    }

  }

  void signInUser({

    required BuildContext context,

    required String email,

    required String password,

  }) {}

}

**Now to use this in auth\_screen.dart**

Auth\_screen.dart

//code

class \_AuthScreenState extends State<AuthScreen> {

  //auth services instance

  final AuthServices \_authServices = AuthServices();

//code

  //sign up logic

  void userSignUp() {

    \_authServices.signUpUser(

        context: context,

        email: \_emailController.text,

        password: \_passwordController.text,

        name: \_nameController.text);

  }

//code

 @override

  Widget build(BuildContext context) {

//code

 CustomButton(

text: "Create account",

onTap: () {

if (\_signUpFormKey.currentState!.validate()) {

userSignUp();

}

}),

Now to run app on node server by “npm run dev”

Then run fn+f5 and then now you can sign up and do not forgot to use

String uri = 'http://<port number>:3000';

And passing our url for signup

  Uri.parse('$uri/api/signup'),

1. **Error handling and snack bar:**

**Error\_handling.dart**

import 'dart:convert';

import 'package:amazon/utils/exports.dart';

import 'package:amazon/utils/show\_snakbar.dart';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

void httpErrorHandling({

  required http.Response response,

  required VoidCallback onSuccess,

  required BuildContext context,

}) {

  switch (response.statusCode) {

    case 200:

      onSuccess();

      break;

    case 400:

      showSnackBar(context, jsonDecode(response.body)['msg']);

      break;

    case 500:

      showSnackBar(context, jsonDecode(response.body)['error']);

      break;

    default:

      showSnackBar(context, response.body);

  }

}

**Show\_snackbar.dart**

import 'package:amazon/utils/exports.dart';

void showSnackBar(BuildContext context, String text) {

  ScaffoldMessenger.of(context).showSnackBar(

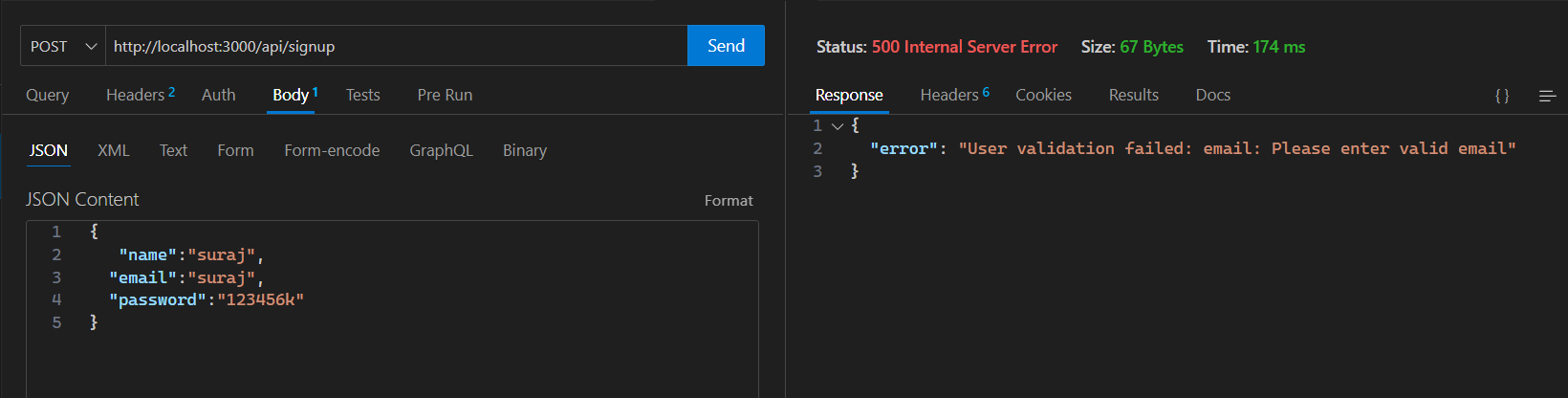
    SnackBar(

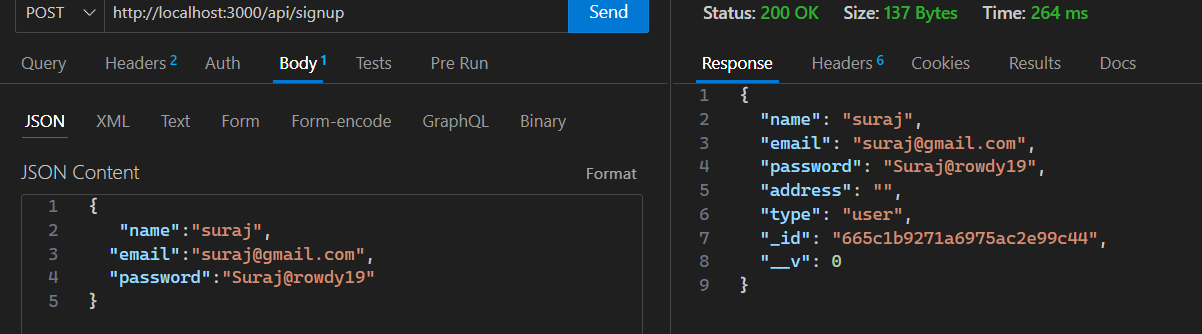
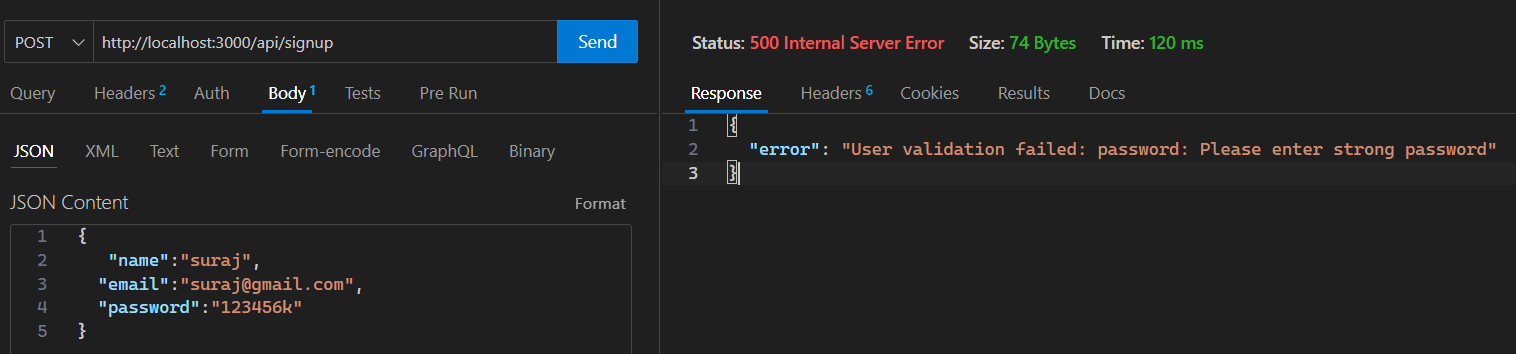
      content: Text(text),

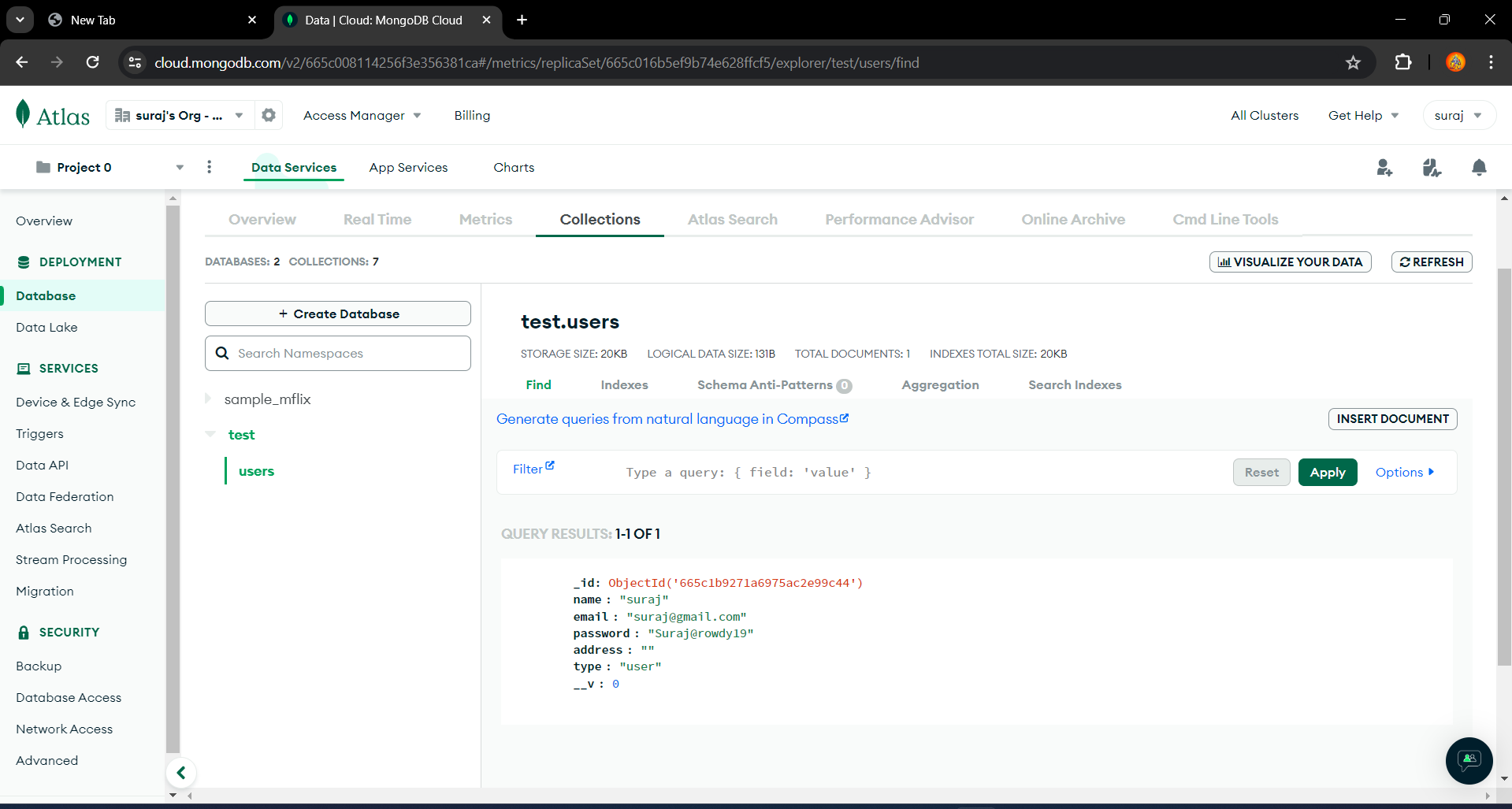
    ),

  );

}

1. **Testing Api “https://localhost:3000/api/signup”-**





1. **To make password secure using bcryptjs –**

/Amazon\_Clone/amazon/server (main)

$ npm i bcryptjs

added 1 package, and audited 116 packages in 3s

auth.js

const bcryptjs = require("bcryptjs");

//code

//add user feilds in model

        //hash password

       const hashedPassword=await bcryptjs.hash(password,8);

        let user = new User({

            email,

            name,

            password:hashedPassword,

        });

//code

1. **For user sign in**

Using jwt(jsonwebtoken) for user sign in on server side

And provider(better state managment),sharedPrefrences (storing token on physical device storage)are using in dart

Auth.js

//code

//sign in user

authRouter.post('/api/signin', async (req, res) => {

    try {

        const { email, password } = req.body;

        const validUser = await User.findOne({ email });

        if (!validUser) {

            return res.status(400).json({ msg: "Email does not exist in database!" });

        }

        //becripting password

        const isMatch = await bcryptjs.compare(password, validUser.password);

        if (!isMatch) {

            return res.status(400).json({ msg: "Incorrect password!" });

        }

        //jwt token

        const token = jwt.sign({ id: validUser.\_id }, "passwordKey");//passwordKey=> secretOrPrivateKey: jwt.Secret

        res.json({ token, ...validUser.\_doc });

    } catch (e) {

        res.status(500).json({ error: e.message });

    }

});

//code

Binding it to dart

Auth\_services.dart

//code

void signInUser({

    required BuildContext context,

    required String email,

    required String password,

  }) async {

    try {

      http.Response res = await http.post(

        Uri.parse('$uri/api/singin'),

        //now just encoding comming data

        body: jsonEncode(

          {

            "email": email,

            'password': password,

          },

        ),

        headers: <String, String>{

          'Content-Type': 'application/json; charset=UTF-8',

        },

      );

      httpErrorHandling(

        response: res,

        context: context,

        onSuccess: () {},

      );

    } catch (e) {

      return showSnackBar(context, e.toString());

    }

  }

//code

Auth\_screen.dart

class AuthScreen extends StatefulWidget {

  //code

}

class \_AuthScreenState extends State<AuthScreen> {

//code

//sign in logic

  void userSignIn() {

    \_authServices.signInUser(

        context: context,

        email: \_emailController.text,

        password: \_passwordController.text);

  }

//code

 @override

  Widget build(BuildContext context) {

//code

 CustomButton(

text: "Sign In",

onTap: () {

if (\_signInFormKey.currentState!.validate()) {

userSignIn();

}

}),

//code

1. **Testing api “https://localhost:3000/api/signin”-**

