Controller

Todo.controller.js

const ToDoService = require('../services/todo.services');

exports.createToDo =  async (req,res,next)=>{

    try {

        const { userId,title, desc } = req.body;

        let todoData = await ToDoService.createToDo(userId,title, desc);

        res.json({status: true,success:todoData});

    } catch (error) {

        console.log(error, 'err---->');

        next(error);

    }

}

exports.getToDoList =  async (req,res,next)=>{

    try {

        const { userId } = req.body;

        let todoData = await ToDoService.getUserToDoList(userId);

        res.json({status: true,success:todoData});

    } catch (error) {

        console.log(error, 'err---->');

        next(error);

    }

}

exports.deleteToDo =  async (req,res,next)=>{

    try {

        const { id } = req.body;

        let deletedData = await ToDoService.deleteToDo(id);

        res.json({status: true,success:deletedData});

    } catch (error) {

        console.log(error, 'err---->');

        next(error);

    }

}

User.controller.js

const UserService = require("../services/user.services");

exports.register = async (req, res, next) => {

    try {

        const { email, password } = req.body;

        const successRes = await UserService.registerUser(email, password);

        res.json({ status: true, successRes: "User registered Succesfully" });

    } catch (error) {

        throw error

    }

}

exports.login = async (req, res, next) => {

    try {

        const { email, password } = req.body;

        console.log("........",password);

        const user = await UserService.checkuser(email);

        console.log("..........user...............",user);

        if(!user){

            throw new Error('User Dont Exist');

        }

        const isMatch = await user.comparePassword(password);

        if(isMatch === false){

            throw new Error('Password Invalid');

        }

        let tokendata = {\_id:user.\_id, email:user.email};

        const token = await UserService.generateToken(tokendata,"secretKey",'1h')

        res.status(200).json({status:true,token:token})

    } catch (error) {

        throw error

    }

}

Model

Todo.model.js

const db = require('../config/db');

const mongoose = require('mongoose');

const UserModel = require("../model/user.model")

const { Schema } = mongoose;

const todoSchema = new Schema({

    userId:{

        type: Schema.Types.ObjectId,

        ref: UserModel.modelName

    },

    title: {

        type: String,

        required: true,

    },

    desc: {

        type: String,

        required: true,

    }

});

const ToDoModel = db.model('todo', todoSchema);

module.exports = ToDoModel;

user.model.js

const mongoose = require('mongoose');

const bcrypt = require("bcrypt");

const db = require('../config/db'); // Adjust the path if necessary

const { Schema } = mongoose;

const userSchema = new Schema({

    email: {

        type: String,

        lowercase: true,

        required: true,

        unique: true,

    },

    password: {

        type: String,

        required: true,

    }

});

userSchema.pre('save',async function(){

    try {

        var user = this;

        const salt = await(bcrypt.genSalt(10));

        const hashpass = await bcrypt.hash(user.password,salt);

        user.password = hashpass;

    } catch(error) {

        throw error;

    }

});

userSchema.methods.comparePassword = async function(userPassword) {

    try{

        console.log('-------------no password',this.password);

        const isMatch = await bcrypt.compare(userPassword, this.password);

        return isMatch;

    }catch (error){

        throw error;

    }

}

const UserModel = db.model('user', userSchema);

module.exports = UserModel;

routers

todo.router.js

const router = require("express").Router();

const ToDoController = require('../controller/todo.controller')

router.post("/createTodo",ToDoController.createToDo);

router.get('/getUserTodoList',ToDoController.getToDoList)

router.post("/deleteTodo",ToDoController.deleteToDo)

module.exports = router;

user.router.js

const router = require('express').Router();

const UserController = require("../controller/user.controller");

router.post('/registration',UserController.register);

router.post('/login',UserController.login);

module.exports = router;

services

todo.services.js

const { deleteToDo } = require("../controller/todo.controller");

const ToDoModel = require("../model/todo.model");

class ToDoService{

    static async createToDo(userId,title,desc){

            const createToDo = new ToDoModel({userId,title,desc});

            return await createToDo.save();

    }

    static async getUserToDoList(userId){

        const todoList = await ToDoModel.find({userId})

        return todoList;

    }

   static async deleteToDo(id){

        const deleted = await ToDoModel.findByIdAndDelete({\_id:id})

        return deleted;

   }

}

module.exports = ToDoService;

user.services.js

const UserModel = require('../model/user.model');

const jwt = require('jsonwebtoken');

class UserService {

    static async registerUser(email, password) {

        try {

            const createUser = new UserModel({ email, password });

            return await createUser.save();

        } catch (err) {

            throw err;

        }

    }

    static async checkuser(email) {

        try {

            return await UserModel.findOne({email});

        } catch (error) {

            throw error;

        }

    }

static async generateToken(tokenData,secretKey,jwt\_expire) {

    return jwt.sign(tokenData,secretKey,{expiresIn:jwt\_expire});

}

}

module.exports = UserService;

app.js

const express = require("express");

const bodyParser = require("body-parser")

const UserRoute = require("./routers/user.router");

const ToDoRoute = require('./routers/todo.router');

const app = express();

app.use(bodyParser.json())

app.use("/",UserRoute);

app.use("/",ToDoRoute);

module.exports = app;

index.js

const app = require('./app');

const db = require('./config/db'); // Import db first

const UserModel = require('./model/user.model'); // Then import UserModel

const ToDoModel = require('./model/todo.model');

const port = 8000;

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

    res.send("Hello World..!");

});

db.on('open', () => {

    // The database connection is open, so you can safely use UserModel here

    app.listen(port, () => {

        console.log(`Server Listening on port http://localhost:${port}`);

    });

}).on('error', (error) => {

    console.error(`Mongodb connection error: ${error}`);

});

**Flutter\_Frontend**

Applogo.dart

 import 'package:flutter/material.dart';

 import 'package:velocity\_x/velocity\_x.dart';

 class CommonLogo extends StatelessWidget {

   @override

   Widget build(BuildContext context) {

     return Column(

       mainAxisAlignment: MainAxisAlignment.center,

       children: <Widget>[

         Image.network("https://pluspng.com/img-png/avengers-logo-png-avengers-logo-png-1376.png",width: 100,),

         "To-Do App".text.xl2.italic.make(),

         "Make A List of your task".text.light.white.wider.lg.make(),

       ],

     );

   }

 }

Config.dart

final url = 'http:// 10.121.46.74:8000/';

final registration = url + "registration";

final login = url + 'login';

final addtodo = url + 'createTodo';

final getToDoList = url + 'getUserTodoList';

final deleteTodo = url + 'deleteTodo';

dashboard.dart

import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:jwt\_decoder/jwt\_decoder.dart';

import 'package:velocity\_x/velocity\_x.dart';

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

import 'config.dart';

import 'package:flutter\_slidable/flutter\_slidable.dart';

class Dashboard extends StatefulWidget {

  final token;

  const Dashboard({@required this.token,Key? key}) : super(key: key);

  @override

  State<Dashboard> createState() => \_DashboardState();

}

class \_DashboardState extends State<Dashboard> {

  late String userId;

  TextEditingController \_todoTitle = TextEditingController();

  TextEditingController \_todoDesc = TextEditingController();

  List? items;

  @override

  void initState() {

    // TODO: implement initState

    super.initState();

    Map<String,dynamic> jwtDecodedToken = JwtDecoder.decode(widget.token);

    userId = jwtDecodedToken['\_id'];//id which is passed in controller.js

    getTodoList(userId);

  }

  void addTodo() async{

    if(\_todoTitle.text.isNotEmpty && \_todoDesc.text.isNotEmpty){

      var regBody = {

        "userId":userId,

        "title":\_todoTitle.text,

        "desc":\_todoDesc.text

      };

      var response = await http.post(Uri.parse(addtodo),

          headers: {"Content-Type":"application/json"},

          body: jsonEncode(regBody)

      );

      var jsonResponse = jsonDecode(response.body);

      print(jsonResponse['status']);

      if(jsonResponse['status']){

        \_todoDesc.clear();

        \_todoTitle.clear();

        Navigator.pop(context);

        getTodoList(userId);

      }else{

        print("SomeThing Went Wrong");

      }

    }

  }

  void getTodoList(userId) async {

    var regBody = {

      "userId":userId

    };

    var response = await http.post(Uri.parse(getToDoList),

        headers: {"Content-Type":"application/json"},

        body: jsonEncode(regBody)

    );

    var jsonResponse = jsonDecode(response.body);

    items = jsonResponse['success'];

    setState(() {

    });

  }

  void deleteItem(id) async{

    var regBody = {

      "id":id

    };

    var response = await http.post(Uri.parse(deleteTodo),

        headers: {"Content-Type":"application/json"},

        body: jsonEncode(regBody)

    );

    var jsonResponse = jsonDecode(response.body);

    if(jsonResponse['status']){

      getTodoList(userId);

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: Colors.lightBlueAccent,

       body: Column(

         crossAxisAlignment: CrossAxisAlignment.start,

         children: [

           Container(

             padding: EdgeInsets.only(top: 60.0,left: 30.0,right: 30.0,bottom: 30.0),

             child: Column(

               crossAxisAlignment: CrossAxisAlignment.start,

               children: [

                 CircleAvatar(child: Icon(Icons.list,size: 30.0,),backgroundColor: Colors.white,radius: 30.0,),

                 SizedBox(height: 10.0),

                 Text('ToDo with NodeJS + Mongodb',style: TextStyle(fontSize: 30.0,fontWeight: FontWeight.w700),),

                 SizedBox(height: 8.0),

                 Text('5 Task',style: TextStyle(fontSize: 20),),

               ],

             ),

           ),

           Expanded(

             child: Container(

               decoration: BoxDecoration(

                   color: Colors.white,

                   borderRadius: BorderRadius.only(topLeft: Radius.circular(20),topRight: Radius.circular(20))

               ),

               child: Padding(

                 padding: const EdgeInsets.all(8.0),

                 child: items == null ? null : ListView.builder(

                     itemCount: items!.length,

                     itemBuilder: (context,int index){

                       return Slidable(

                         key: const ValueKey(0),

                         endActionPane: ActionPane(

                           motion: const ScrollMotion(),

                           dismissible: DismissiblePane(onDismissed: () {}),

                           children: [

                             SlidableAction(

                               backgroundColor: Color(0xFFFE4A49),

                               foregroundColor: Colors.white,

                               icon: Icons.delete,

                               label: 'Delete',

                               onPressed: (BuildContext context) {

                                 print('${items![index]['\_id']}');

                                 deleteItem('${items![index]['\_id']}');

                               },

                             ),

                           ],

                         ),

                         child: Card(

                           borderOnForeground: false,

                           child: ListTile(

                             leading: Icon(Icons.task),

                             title: Text('${items![index]['title']}'),

                             subtitle: Text('${items![index]['desc']}'),

                             trailing: Icon(Icons.arrow\_back),

                           ),

                         ),

                       );

                     }

                 ),

               ),

             ),

           )

         ],

       ),

      floatingActionButton: FloatingActionButton(

        onPressed: () =>\_displayTextInputDialog(context) ,

        child: Icon(Icons.add),

        tooltip: 'Add-ToDo',

      ),

    );

  }

  Future<void> \_displayTextInputDialog(BuildContext context) async {

    return showDialog(

        context: context,

        builder: (context) {

          return AlertDialog(

            title: Text('Add To-Do'),

            content: Column(

              mainAxisSize: MainAxisSize.min,

              children: [

                TextField(

                  controller: \_todoTitle,

                  keyboardType: TextInputType.text,

                  decoration: InputDecoration(

                      filled: true,

                      fillColor: Colors.white,

                      hintText: "Title",

                      border: OutlineInputBorder(

                          borderRadius: BorderRadius.all(Radius.circular(10.0)))),

                ).p4().px8(),

                TextField(

                  controller: \_todoDesc,

                  keyboardType: TextInputType.text,

                  decoration: InputDecoration(

                      filled: true,

                      fillColor: Colors.white,

                      hintText: "Description",

                      border: OutlineInputBorder(

                          borderRadius: BorderRadius.all(Radius.circular(10.0)))),

                ).p4().px8(),

                ElevatedButton(onPressed: (){

                  addTodo();

                  }, child: Text("Add"))

              ],

            )

          );

        });

  }

}

loginPage.dart

import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:flutter\_todo\_app/dashboard.dart';

import 'package:flutter\_todo\_app/registration.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'package:velocity\_x/velocity\_x.dart';

import 'applogo.dart';

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

import 'config.dart';

class SignInPage extends StatefulWidget {

  @override

  \_SignInPageState createState() => \_SignInPageState();

}

class \_SignInPageState extends State<SignInPage> {

  TextEditingController emailController = TextEditingController();

  TextEditingController passwordController = TextEditingController();

  bool \_isNotValidate = false;

  late SharedPreferences prefs;

  @override

  void initState() {

    // TODO: implement initState

    super.initState();

    initSharedPref();

  }

  void initSharedPref() async{

    prefs = await SharedPreferences.getInstance();

  }

  void loginUser() async{

    if(emailController.text.isNotEmpty && passwordController.text.isNotEmpty){

      var reqBody = {

        "email":emailController.text,

        "password":passwordController.text

      };

      var response = await http.post(Uri.parse(registration),

          headers: {"Content-Type":"application/json"},

          body: jsonEncode(reqBody)

      );

      var jsonResponse = jsonDecode(response.body);

      if(jsonResponse['status']){

          var myToken = jsonResponse['token'];

          prefs.setString('token', myToken);

          Navigator.push(context, MaterialPageRoute(builder: (context)=>Dashboard(token: myToken)));

      }else{

        print('Something went wrong');

      }

    }

  }

  @override

  Widget build(BuildContext context) {

    return SafeArea(

      child: Scaffold(

        body: Container(

          width: MediaQuery.of(context).size.width,

          height: MediaQuery.of(context).size.height,

          decoration: BoxDecoration(

            gradient: LinearGradient(

                colors: [const Color(0XFFF95A3B),const Color(0XFFF96713)],

                begin: FractionalOffset.topLeft,

                end: FractionalOffset.bottomCenter,

                stops: [0.0,0.8],

                tileMode: TileMode.mirror

            ),

          ),

          child: Center(

            child: SingleChildScrollView(

              child: Column(

                mainAxisAlignment: MainAxisAlignment.center,

                children: <Widget>[

                  CommonLogo(),

                  HeightBox(10),

                  "Email Sign-In".text.size(22).yellow100.make(),

                  TextField(

                    controller: emailController,

                    keyboardType: TextInputType.text,

                    decoration: InputDecoration(

                        filled: true,

                        fillColor: Colors.white,

                        hintText: "Email",

                        errorText: \_isNotValidate ? "Enter Proper Info" : null,

                        border: OutlineInputBorder(

                            borderRadius: BorderRadius.all(Radius.circular(10.0)))),

                  ).p4().px24(),

                  TextField(

                    controller: passwordController,

                    keyboardType: TextInputType.text,

                    decoration: InputDecoration(

                        filled: true,

                        fillColor: Colors.white,

                        hintText: "Password",

                        errorText: \_isNotValidate ? "Enter Proper Info" : null,

                        border: OutlineInputBorder(

                            borderRadius: BorderRadius.all(Radius.circular(10.0)))),

                  ).p4().px24(),

                  GestureDetector(

                    onTap: (){

                        loginUser();

                    },

                    child: HStack([

                      VxBox(child: "LogIn".text.white.makeCentered().p16()).green600.roundedLg.make(),

                    ]),

                  ),

                ],

              ),

            ),

          ),

        ),

        bottomNavigationBar: GestureDetector(

          onTap: (){

            Navigator.push(context, MaterialPageRoute(builder: (context)=>Registration()));

          },

          child: Container(

              height: 25,

              color: Colors.lightBlue,

              child: Center(child: "Create a new Account..! Sign Up".text.white.makeCentered())),

        ),

      ),

    );

  }

}

Main.dart

import 'package:flutter/material.dart';

import 'package:velocity\_x/velocity\_x.dart';

import 'package:flutter\_todo\_app/dashboard.dart';

import 'package:jwt\_decoder/jwt\_decoder.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'loginPage.dart';

import 'applogo.dart';

import 'registration.dart';

void main() async {

  WidgetsFlutterBinding.ensureInitialized();

  // ignore: unused\_local\_variable

  SharedPreferences prefs = await SharedPreferences.getInstance();

  runApp(MyApp(token: prefs.getString('token'),));

}

class MyApp extends StatelessWidget {

  final token;

  const MyApp({

    @required this.token,

    Key? key,

  }) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Flutter Demo',

      debugShowCheckedModeBanner: false,

      theme: ThemeData(

        primaryColor: Colors.black,

        visualDensity: VisualDensity.adaptivePlatformDensity,

      ),

      home: SignInPage(),

    );

  }

}

Registration.dart

import 'dart:convert';

import 'dart:math';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:velocity\_x/velocity\_x.dart';

import 'applogo.dart';

import 'loginPage.dart';

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

import 'config.dart';

class Registration extends StatefulWidget {

  @override

  \_RegistrationState createState() => \_RegistrationState();

}

class \_RegistrationState extends State<Registration> {

  TextEditingController emailController = TextEditingController();

  TextEditingController passwordController = TextEditingController();

  bool \_isNotValidate = false;

  void registerUser() async{

    if(emailController.text.isNotEmpty && passwordController.text.isNotEmpty){

      var regBody = {

        "email":emailController.text,

        "password":passwordController.text

      };

      var response = await http.post(Uri.parse(registration),

      headers: {"Content-Type":"application/json"},

      body: jsonEncode(regBody)

      );

      var jsonResponse = jsonDecode(response.body);

      print(jsonResponse['status']);

      if(jsonResponse['status']){

        Navigator.push(context, MaterialPageRoute(builder: (context)=>SignInPage()));

      }else{

        print("SomeThing Went Wrong");

      }

    }else{

      setState(() {

        \_isNotValidate = true;

      });

    }

  }

  @override

  Widget build(BuildContext context) {

    return SafeArea(

      child: Scaffold(

        body: Container(

          width: MediaQuery.of(context).size.width,

          height: MediaQuery.of(context).size.height,

          decoration: BoxDecoration(

            gradient: LinearGradient(

                colors: [const Color(0XFFF95A3B),const Color(0XFFF96713)],

                begin: FractionalOffset.topLeft,

                end: FractionalOffset.bottomCenter,

                stops: [0.0,0.8],

                tileMode: TileMode.mirror

            ),

          ),

          child: Center(

            child: SingleChildScrollView(

              child: Column(

                mainAxisAlignment: MainAxisAlignment.center,

                children: <Widget>[

                  CommonLogo(),

                  HeightBox(10),

                  "CREATE YOUR ACCOUNT".text.size(22).yellow100.make(),

                  TextField(

                    controller: emailController,

                    keyboardType: TextInputType.text,

                    decoration: InputDecoration(

                        filled: true,

                        fillColor: Colors.white,

                        errorStyle: TextStyle(color: Colors.white),

                        errorText: \_isNotValidate ? "Enter Proper Info" : null,

                        hintText: "Email",

                        border: OutlineInputBorder(

                            borderRadius: BorderRadius.all(Radius.circular(10.0)))),

                  ).p4().px24(),

                  TextField(

                    controller: passwordController,

                    keyboardType: TextInputType.text,

                    decoration: InputDecoration(

                        suffixIcon: IconButton(icon: Icon(Icons.copy),onPressed: (){

                          final data = ClipboardData(text: passwordController.text);

                          Clipboard.setData(data);

                        },),

                        prefixIcon: IconButton(icon: Icon(Icons.password),onPressed: (){

                          String passGen =  generatePassword();

                          passwordController.text = passGen;

                          setState(() {

                          });

                        },),

                        filled: true,

                        fillColor: Colors.white,

                        errorStyle: TextStyle(color: Colors.white),

                        errorText: \_isNotValidate ? "Enter Proper Info" : null,

                        hintText: "Password",

                        border: OutlineInputBorder(

                            borderRadius: BorderRadius.all(Radius.circular(10.0)))),

                  ).p4().px24(),

                  HStack([

                    GestureDetector(

                      onTap: ()=>{

                        registerUser()

                      },

                        child: VxBox(child: "Register".text.white.makeCentered().p16()).green600.roundedLg.make().px16().py16(),

                    ),

                  ]),

                  GestureDetector(

                    onTap: (){

                      print("Sign In");

                      Navigator.push(context, MaterialPageRoute(builder: (context)=>SignInPage()));

                    },

                    child: HStack([

                      "Already Registered?".text.make(),

                      " Sign In".text.white.make()

                    ]).centered(),

                  )

                ],

              ),

            ),

          ),

        ),

      ),

    );

  }

}

String generatePassword() {

  String upper = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

  String lower = 'abcdefghijklmnopqrstuvwxyz';

  String numbers = '1234567890';

  String symbols = '!@#\$%^&\*()<>,./';

  String password = '';

  int passLength = 20;

  String seed = upper + lower + numbers + symbols;

  List<String> list = seed.split('').toList();

  Random rand = Random();

  for (int i = 0; i < passLength; i++) {

    int index = rand.nextInt(list.length);

    password += list[index];

  }

  return password;

}