**Full Stack Development with MERN**

**Database Design and Development Report**

|  |  |
| --- | --- |
| Date | 16/07/2024 |
| Team ID | SWTID1720074953 |
| Project Name | SB FOODS-FOOD ORDERING APP |
| Maximum Marks |  |

**Project Title**: SB FOODS-FOOD ORDERING APP

**Date**: 16/07/2024

**Prepared by**: K.Bhavana

Bodagala Gayathri

K Bhuvaneswari

P Lahari

**Objective**

The objective of this report is to outline the database design and implementation details for the Food Delivery Web Application project, including schema design and database management system (DBMS) integration. This will cover how data is structured, stored, and managed to ensure efficient and reliable operations of the application.

**Technologies Used**

**MongoDB (Database Management System)**

MongoDB is like a super flexible, powerful filing cabinet for your app. Instead of organizing data in neat rows and columns like traditional databases, it stores information in a more free-form, JSON-like format. This makes it perfect for a food delivery app, where you need to store a variety of information, like user profiles, restaurant details, menus, orders, and delivery updates. It's great at handling lots of data quickly and can easily adapt to changes.

**Mongoose (Object-Document Mapper)**

Mongoose is a helpful tool that acts as a bridge between MongoDB and your application. Think of it as a translator that helps your app speak MongoDB's language. It allows you to define what your data should look like and ensures that the data fits the expected structure. For a food delivery app, Mongoose makes it easier to manage complex data, like user accounts, restaurant menus, and order details, without having to write a lot of complicated code. It keeps everything organized and reduces the chance of making mistakes.

**Design the Database Schema**

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

**1. Users**

- Attributes: [list attributes like \_id, name, email, password, createdAt, updatedAt]

**2. Posts**

- Attributes: [list attributes like \_id, title, content, author (references User), createdAt, updatedAt]

**3. Comments**

- Attributes: [list attributes like \_id, text, post (references Post), author (references User), createdAt, updatedAt]

**Implement the Database using MongoDB**

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

**Database Name:** [db.js]

import mongoose from "mongoose";

export constconnectDB = async () => {

    await mongoose.connect('mongodb+srv://Gayathri:Gundu2003@cluster0.tiebsgz.mongodb.net/food-del').then(() => console.log("DB Connected"));

}

**Index.js**

'use strict';

var whichBoxedPrimitive = require('which-boxed-primitive');

var whichCollection = require('which-collection');

var whichTypedArray = require('which-typed-array');

var isArray = require('isarray');

var isDate = require('is-date-object');

var isRegex = require('is-regex');

var isWeakRef = require('is-weakref');

var isFinalizationRegistry = require('is-finalizationregistry');

var name = require('function.prototype.name');

var isGeneratorFunction = require('is-generator-function');

var isAsyncFunction = require('is-async-function');

var hasToStringTag = require('has-tostringtag/shams')();

var toStringTag = hasToStringTag&&Symbol.toStringTag;

var $Object = Object;

var promiseThen = typeof Promise === 'function' &&Promise.prototype.then;

var isPromise = function isPromise(value) {

    if (!value || typeof value !== 'object' || !promiseThen) {

        return false;

    }

    try {

        promiseThen.call(value, null, function () {});

        return true;

    } catch (e) {}

    return false;

};

var isKnownBuiltin = function isKnownBuiltin(builtinName) {

    return builtinName

        // primitives

        &&builtinName !== 'BigInt'

        &&builtinName !== 'Boolean'

        &&builtinName !== 'Null'

        &&builtinName !== 'Number'

        &&builtinName !== 'String'

        &&builtinName !== 'Symbol'

        &&builtinName !== 'Undefined'

        // namespaces

        &&builtinName !== 'Math'

        &&builtinName !== 'JSON'

        &&builtinName !== 'Reflect'

        &&builtinName !== 'Atomics'

        // collections

        &&builtinName !== 'Map'

        &&builtinName !== 'Set'

        &&builtinName !== 'WeakMap'

        &&builtinName !== 'WeakSet'

        // typed arrays

        &&builtinName !== 'BigInt64Array'

        &&builtinName !== 'BigUint64Array'

        &&builtinName !== 'Float32Array'

        &&builtinName !== 'Float64Array'

        &&builtinName !== 'Int16Array'

        &&builtinName !== 'Int32Array'

        &&builtinName !== 'Int8Array'

        &&builtinName !== 'Uint16Array'

        &&builtinName !== 'Uint32Array'

        &&builtinName !== 'Uint8Array'

        &&builtinName !== 'Uint8ClampedArray'

        // checked explicitly

        &&builtinName !== 'Array'

        &&builtinName !== 'Date'

        &&builtinName !== 'FinalizationRegistry'

        &&builtinName !== 'Promise'

        &&builtinName !== 'RegExp'

        &&builtinName !== 'WeakRef'

        // functions

        &&builtinName !== 'Function'

        &&builtinName !== 'GeneratorFunction'

        &&builtinName !== 'AsyncFunction';

};

module.exports = function whichBuiltinType(value) {

    if (value == null) {

        return value;

    }

    // covers: primitives, {,Weak}Map/Set, typed arrays

    var which = whichBoxedPrimitive($Object(value)) || whichCollection(value) || whichTypedArray(value);

    if (which) {

        return which;

    }

    if (isArray(value)) {

        return 'Array';

    }

    if (isDate(value)) {

        return 'Date';

    }

    if (isRegex(value)) {

        return 'RegExp';

    }

    if (isWeakRef(value)) {

        return 'WeakRef';

    }

    if (isFinalizationRegistry(value)) {

        return 'FinalizationRegistry';

    }

    if (typeof value === 'function') {

        if (isGeneratorFunction(value)) {

            return 'GeneratorFunction';

        }

        if (isAsyncFunction(value)) {

            return 'AsyncFunction';

        }

        return 'Function';

    }

    if (isPromise(value)) {

        return 'Promise';

    }

    if (toStringTag&&toStringTag in value) {

        var tag = value[toStringTag];

        if (isKnownBuiltin(tag)) {

            return tag;

        }

    }

    if (typeofvalue.constructor === 'function') {

        var constructorName = name(value.constructor);

        if (isKnownBuiltin(constructorName)) {

            return constructorName;

        }

    }

    return 'Object';

};

1. Collection: foods

- Schema:

{"\_id":{"$oid":"669bf955e34fb94c3632cd1f"},"name":"Cucumber","description":"Healthy and energetic!","price":{"$numberInt":"220"},"image":"1720087827645food\_1.png","category":"Salad","\_\_v":{"$numberInt":"0"}}

2. Collection: orders

- Schema:

{"\_id":{"$oid":"668679477f93f6e5c3fa8dc2"},"userId":"6686757384d5afbbd0524209","items":[{"\_id":"6686751384d5afbbd0524204","name":"Cucumber","description":"Healthy and energetic!","price":{"$numberInt":"220"},"image":"1720087827645food\_1.png","category":"Salad","\_\_v":{"$numberInt":"0"},"quantity":{"$numberInt":"1"}}],"amount":{"$numberInt":"270"},"address":{"firstName":"K","lastName":"Bhavana","email":"bhavana.k2021@vitstudent.ac.in","street":"VIT","city":"vellore","state":"TamilNadu","zipcode":"517414","country":"India","phone":"7732066318"},"status":"Out for delivery","date":{"$date":{"$numberLong":"1720088885373"}},"payment":true,"\_\_v":{"$numberInt":"0"}}

3. Collection: users

{"\_id":{"$oid":"6686757384d5afbbd0524209"},"name":"Bhavana","email":"bhavana.k2021@vitstudent.ac.in","password":"$2b$10$iAA74hujIzBEeuXpq3yYA.Gg9XsZJ5UuiqCqePpXczTVgIVMTN6fy","cartData":{},"\_\_v":{"$numberInt":"0"}}

**Integration with Backend**

* Database connection: 