Employee Happiness App

**Abstract**:

Employee happiness survey can help them feel heard and valued by their employer, which can remove bias, improves the wellness decisions and increase morale and job satisfaction.

For employers this kind of survey can provide valuable insights into already provided benefits and what areas need improvement within the organization. Helps employer make informed decisions about how to support their employees and improve the overall work environment.

With this kind of survey Employers can make employees are more productive at work by understanding frequent needs, and engaged in their work, which can lead to increased business value generated. Conversely, if you have unhappy people in your organization, you'll tank profitability. It's always beneficial to keep a finger on this pulse, and internal surveys like this can help.

* Link to Angular app GitHub repository:

<https://github.com/gangaiah548/JavaProjects/tree/main/Angular/employee-Happiness-Survey-api>

* Link to Express API GitHub repository:

<https://github.com/gangaiah548/JavaProjects/tree/main/Angular/employee-Happiness-Survey-api>

**Used technologies:**

* **Angular -15(**Install Angular related env https://angular.io/guide/setup-local)
* **Mongo DB(**Install **Mongo** DB <https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/>)
* **MS Visual Studio**

**Run the app**

Using below command in NPM terminal

npm run start

If you've followed these steps, you will have the following output:

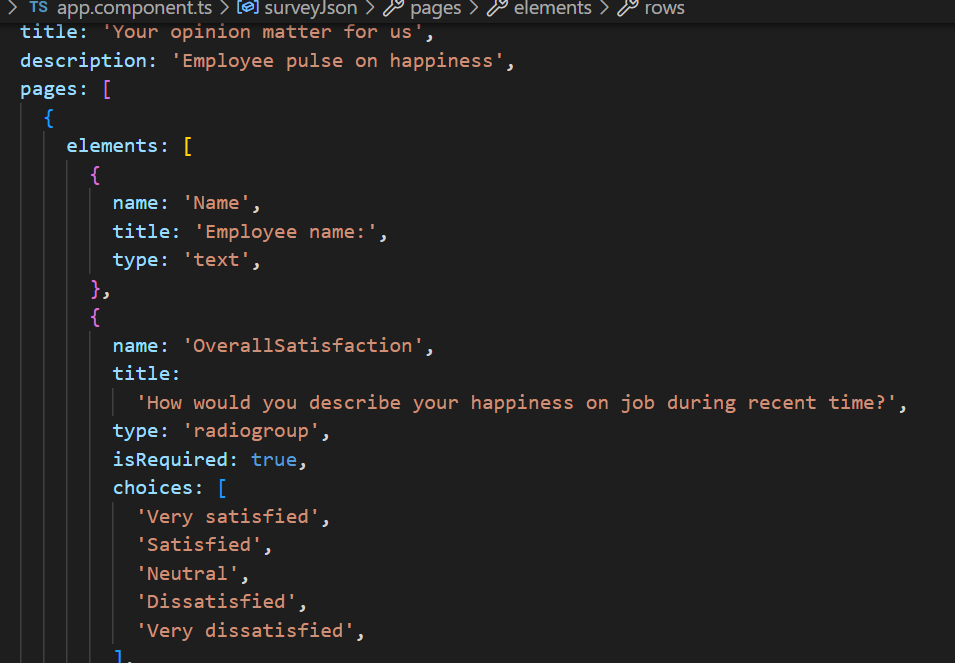


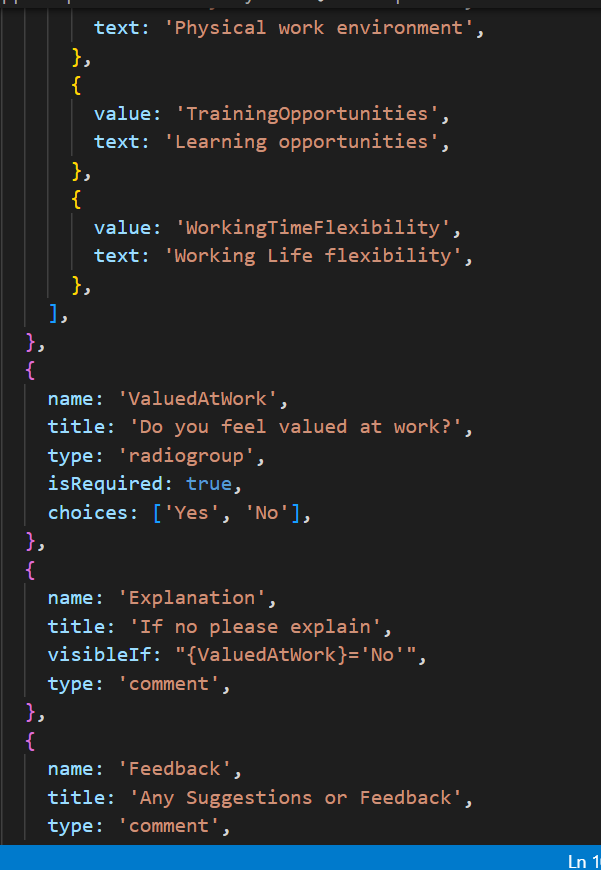
**Create the Employee Happiness Survey**

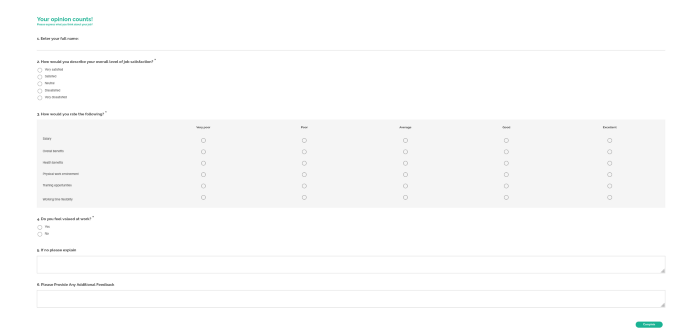
If we take look at our happiness survey, we see that we need 4 types of components:

Employee pulse on happiness

* Simple text field: text (
* Simple text area field: comment
* Radio group field: radiogroup
* Matrix of radio groups: matrix







In order to handle the survey completion after a user completes it, we should use the [onComplete](https://surveyjs.io/Documentation/Library?id=surveymodel&utm_source=medium&utm_medium=referral&utm_campaign=JS_in_Plain_English_8#onComplete) event handler. For example, let's just log (print) the results in the browser console.

Add the following code to your **app.component.ts:**

import { Component, OnInit } from "@angular/core";

import { Model, StylesManager } from "survey-core";

// const SURVEY\_ID = 1;

StylesManager.applyTheme("modern");

const surveyJson = {

// ...

};

@Component({

selector: "app-root",

templateUrl: "./app.component.html",

styleUrls: ["./app.component.css"],

})

export class AppComponent implements OnInit {

title = "My First Survey";

surveyModel: Model;

logSurveyResults(sender) {

console.log(sender.data);

}

ngOnInit() {

const survey = new Model(surveyJson);

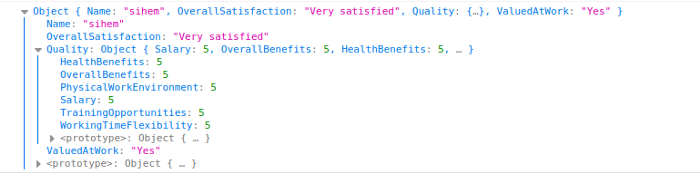
survey.onComplete.add(this.logSurveyResults);

this.surveyModel = survey;

}

}

Save, refresh, and open the browser console to see the results displayed there after completing the survey:



#### Set up the MongoDB Database

Download from <https://www.mongodb.com/docs/manual/installation/>

install node&npm and type below commands in npm

$ node -v

$ npm -v

$ mongo -v

Then, we create our Node.js application and install the required packages

[Express](https://expressjs.com/): A minimal Node.js web application framework that provides a set of features for web and mobile applications. For this type below commands

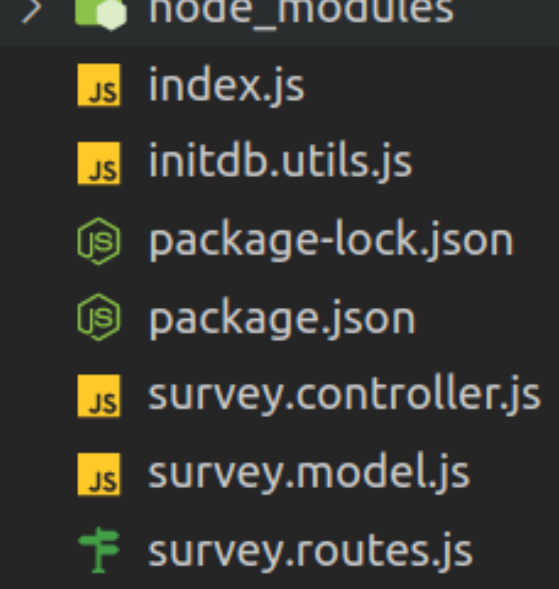
$ mkdir employee-satisfaction-survey-api

$ cd employee-satisfaction-survey-api

$ npm init

$ npm install express nodemon mongoose --- save

Now project structure



**API project structure**

* package.json:

|  |
| --- |
| * { |
|  |

|  |
| --- |
| "name": "employee-Happiness-survey-api", |
|  |

|  |
| --- |
| "version": "1.0.0", |
|  |

|  |
| --- |
| "description": "Employee Happiness survey api", |
|  |

|  |
| --- |
| "main": "index.js", |
|  |

|  |
| --- |
| "scripts": { |
|  |

|  |
| --- |
| "start": "nodemon index.js" |
|  |

|  |
| --- |
| }, |
|  |

|  |
| --- |
| "author": "Gangaiah Lakkakula", |
|  |

|  |
| --- |
| "dependencies": { |
|  |

|  |
| --- |
| "express": "^4.18.2", |
|  |

|  |
| --- |
| "mongoose": "^6.7.4", |
|  |

|  |
| --- |
| "nodemon": "^2.0.20" |
|  |

|  |
| --- |
| } |
|  |

}

* **index.js: The entry point of our API, it contains the Express app and routes declaration:**

// Import dependencies

const express = require("express");

const database = require("./initdb.utils");

// Create app instance

const app = express();

// Define JSON as return type

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

// Configure headers

app.use((req, res, next) => {

res.setHeader("Access-Control-Allow-Origin", "\*");

res.setHeader("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE");

res.setHeader("Access-Control-Allow-Headers", "Content-Type");

res.setHeader("Access-Control-Allow-Credentials", true);

next();

});

// DEFINE ROUTES

app.use("/hello", (req, res) => {

res.status(200).json({ message: "Hello world !" });

});

app.use("/surveys", require("./survey.routes"));

app.all("\*", (req, res) => {

res.status(404).json({ error: "End point not found" });

});

// Handle database error

database.on("error", (error) => {

console.log(

"Connection error: --- --- --- --- --- --- --- --- --- --- --- --- --- "

);

console.log(error);

console.log(

" --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- "

);

});

// Start app after connecting to Database

database.once("connected", () => {

console.log("Database Connected");

const PORT = 3001;

app.listen(PORT, () => console.log("Server ready at 3001"));

});

**initdb.js: Contains connection configuration with our database**

const mongoose = require("mongoose");

const MONGO\_HOST = "localhost";

const MONGO\_PORT = 27017;

const MONGO\_DB\_NAME = "surveydb";

const MONGO\_URL = `mongodb://${MONGO\_HOST}:${MONGO\_PORT}/${MONGO\_DB\_NAME}`;

const connectOptions = {

useNewUrlParser: true,

};

mongoose.connect(MONGO\_URL, connectOptions);

const database = mongoose.connection;

module.exports = database;

**survey.model.js: Contains our survey model schema**

const mongoose = require("mongoose");

const SurveySchema = mongoose.Schema(

{

Name: String,

OverallSatisfaction: String,

Quality: Array,

ValuedAtWork: String,

Explanation: String,

Feedback: String,

},

{

timestamps: true,

strict: false,

}

);

module.exports = mongoose.model("Survey", SurveySchema);

* **survey.controller.js:** Contains our web services, in our example, we will create only two services one for saving the results of the survey and another for retrieving results.

const Survey = require("./survey.model");

// Create and Save a new Survey

exports.create = (req, res) => {

const surveyData = req.body;

// Create a Post

const survey = new Survey(surveyData);

// Save Post in the database

survey

.save()

.then((data) => {

res.send(data);

})

.catch((err) => {

res.status(500).send({

message:

err.message || "Some error occurred while creating the survey.",

});

});

};

// Find all surveys

exports.findAll = (req, res) => {

Survey.find({})

.then((data) => {

res.send(data);

})

.catch((err) => {

res.status(500).send({

message: err.message || "Some error occurred while retrieving data.",

});

});

};

* **survey.routes.js: Contains the routes of our API**

const surveyController = require("./survey.controller");

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

//CRUD

router.post("/", surveyController.create).get("/", surveyController.findAll);

module.exports = router;

**Now let's start our API by running the following command:**

npm run start

If you go to <http://localhost:3001/hello> you will see the message "hello world!"

If you try <http://localhost:3001/surveys> you will get an empty response because there is no survey saved yet.

In the next section, we will save the results by using our API.

#### Store Results in MongoDB Collection

Now that we have our API, let's update the app.component.ts file, and instead of just logging the survey results, we will send them by an HTTP request to our API.

import { Component, OnInit } from "@angular/core";

import { Model, StylesManager } from "survey-core";

// const SURVEY\_ID = 1;

StylesManager.applyTheme("modern");

const surveyJson = {

// ...

};

@Component({

selector: "app-root",

templateUrl: "./app.component.html",

styleUrls: ["./app.component.css"],

})

export class AppComponent implements OnInit {

title = "My First Survey";

surveyModel: Model;

logSurveyResults(sender) {

console.log(sender.data);

}

saveSurveyResults(sender) {

const request = new XMLHttpRequest();

const url = "<http://localhost:3001/surveys>";

request.open("POST", url);

request.setRequestHeader("Content-Type", "application/json;charset=UTF-8");

request.send(JSON.stringify(sender.data));

}

ngOnInit() {

const survey = new Model(surveyJson);

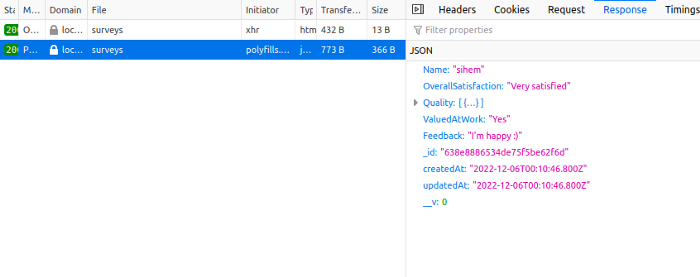
survey.onComplete.add(this.saveSurveyResults);

this.surveyModel = survey;

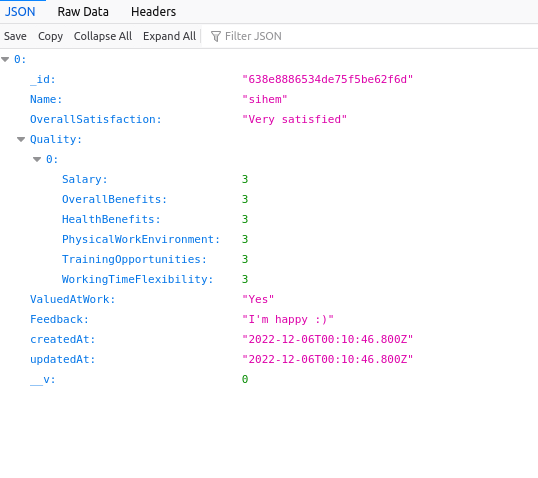
}

}

**After that and after filling all the fields if we click on complete the survey button, data will be sent to the API:**



And, if you try now to go to <http://localhost:3001/surveys>, you will find the results:



With this simple demonstration ending up. Please provide your feedback and any suggestion taking up this survey development further.