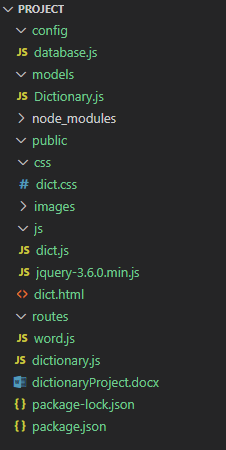
**DICTIONARY PROJECT**

// Project Structure



I first did it the straight forward way of extracting data from MySQL and sending it to the frontend without using **Sequelize**. But, later I decided to use **sequelize** to make it work for easily for other relational databases like PostgreSQL, MS SQL and others in case the database changes.

I have also tried to modularize the project and added models, routes, and db configuration files as is usually done for larger projects.

The main components (folders) of the project are

* Config - contains database configuration file
* Models – contains data models for the provided ‘entries’ database
* Routes – contains all routes for the project
* Public – contains all frontend code for the project including html, css, JS and jQuery code
* Dictionary.js – the NodeJS Express web server

**Packages and Tools Used**

* Express
* Squelize
* Mysql2
* Nodemon
* Postman

// package.json

{

  "name": "wap-online-dictionary",

  "version": "1.0.0",

  "description": "WAP Online Dictionary",

  "main": "dictionary.js",

  "scripts": {

    "start": "nodemon ./dictionary.js",

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

  },

  "author": "Berhanu",

  "license": "ISC",

  "dependencies": {

    "express": "^4.17.1",

    "mysql2": "^2.3.3",

    "sequelize": "^6.9.0"

  }

}

// config/database.js

const Sequelize = require("sequelize");

const DB\_NAME = "entries";

const DB\_USER = "wap";

const DB\_PASSWORD = "miuDBPassw0rd";

module.exports = new Sequelize(DB\_NAME, DB\_USER, DB\_PASSWORD, {

  host: "localhost",

  dialect: "mysql",

  operatorAliases: false,

  pool: {

    max: 5,

    min: 0,

    acquire: 30000,

    idle: 10000,

  },

});

// models/Dictionary.js

const Sequelize = require("sequelize");

const db = require("../config/database");

const Entries = db.define("entries", {

  word: {

    type: Sequelize.STRING,

  },

  wordtype: {

    type: Sequelize.STRING,

  },

  definition: {

    type: Sequelize.STRING,

  },

});

module.exports = Entries;

routes/word.js – Database query based on route code

const express = require("express");

const Dict = require("../models/Dictionary");

// const db = require("../config/database");

const router = express.Router();

router.post("/", (req, res) => {

  Dict.findAll({

    where: {

      word: req.body.word,

    },

    attributes: ["word", "wordtype", "definition"],

  })

    .then((dict) => {

      res.send(dict);

    })

    .catch((err) => {

      console.log(err);

    });

});

module.exports = router;

//dictionary.js – Server code

const express = require("express");

const Sequelize = require("sequelize");

const port = process.env.PORT || 3000;

// Database

const db = require("./config/database");

// Test DB

db.authenticate()

  .then(() => {

    console.log("Successfully Connected to MySQL!");

  })

  .catch((err) => {

    console.log("Error: " + err);

  });

const app = express();

app.use(express.json());

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

app.use(express.static("./public"));

// Home route

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

  res.redirect("dict.html");

});

// Dictionary Route

const dictRoute = require("./routes/word");

app.use("/search", dictRoute);

app.listen(port, () => {

  console.log(`Web server listening on port ${port} ...`);

});

// public/dict.html – frontend code

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>CS472 Online Dictionary</title>

  <script src="./js/jquery-3.6.0.min.js"></script>

  <script src="./js/dict.js"></script>

  <link rel="stylesheet" href="./css/dict.css">

</head>

<body>

  <header>

    <div class="container">

      <img id="header-img" src="./images/miu-logo-horizontal-1024x317.png" alt="">

    </div>

  </header>

  <div class="container">

    <h1>CS472 Online Dictionary</h1>

    <div id="lookup">

      <form action="">

        <label for="search">Term: </label>

        <input type="search" name="word" id="search" minlength="1" placeholder="Search a word" required />

        <input type="submit" id="lookup" value="Lookup">

      </form>

    </div>

    <div id="definition">

    </div>

  </div>

  <footer id="footer">

    <div class="container">

      <div>

        Copyright © 2021 Online Dictionary. All Rights Reserved

      </div>

      <div>

        Maharishi International University

      </div>

      <div><span>Email: </span><a href="mailto:bgdesta@miu.edu">bgdesta@miu.edu </a></div>

    </div>

  </footer>

</body>

</html>

// public/js/dict.js – frontend JS and jQuery code

$(document).ready(function () {

  let url = "http://localhost:3000/search";

  $("form").on("submit", function (event) {

    let searchItem = $(this).serialize();

    $.ajax(url, {

      type: "POST",

      dataType: "json",

      data: searchItem,

    })

      .done(displayMeaning)

      .fail((err) => {

        console.log(err);

      });

    event.preventDefault();

  });

});

let displayMeaning = function (result) {

  $("#definition").html("");

  if (result.length === 0) {

    let notFound = $(

      "<div><p>Sorry, the term you searched for was not found in our Dictionary!</p></div>"

    );

    notFound.css({

      color: "red",

      "font-family": " monospace",

      "font-size": "16px",

      "margin-top": "15px",

    });

    // $("#definition > div").remove();

    $("#definition").prepend(notFound);

  } else {

    $("#definition").append($("<ol></ol>"));

    for (let i = 0; i < result.length; i++) {

      let defn = $(

        "<li>(" + result[i].wordtype + ") :: " + result[i].definition + "</li>"

      );

      defn.appendTo($("#definition > ol"));

    }

  }

};

// public/css/dict.css

body {

  font-family: Arial, Helvetica, sans-serif;

  font-size: 18px;

  line-height: 1.5;

  padding: 0;

  margin: 0;

  overflow: hidden;

}

.container {

  width: 96%;

  margin: auto;

}

/\* Header \*/

header {

  min-height: 70px;

  padding-top: 5px;

  border-bottom: #e8491d 1px solid;

  margin-top: 20px;

}

#header-img {

  min-height: 100%;

  width: 20%;

  background-size: 100% 100%;

  overflow-y: hidden;

  overflow-x: hidden;

}

#lookup {

  font-size: 20px;

  font-family: serif;

}

#search {

  margin-right: 10px;

  height: 30px;

  border-radius: 12px;

}

h1 {

  font-family: "Gill Sans", "Gill Sans MT", sans-serif;

  color: #4a3570;

  margin-top: 10px;

  margin-bottom: 10px;

}

footer {

  position: absolute;

  bottom: 0;

  width: 100%;

  height: 60px;

  font-family: monospace;

  font-size: 14px;

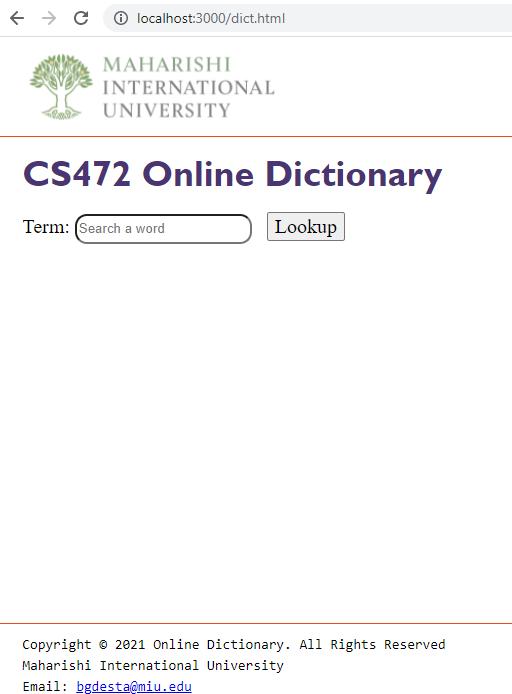
  margin-top: 30px;

  padding-top: 10px;

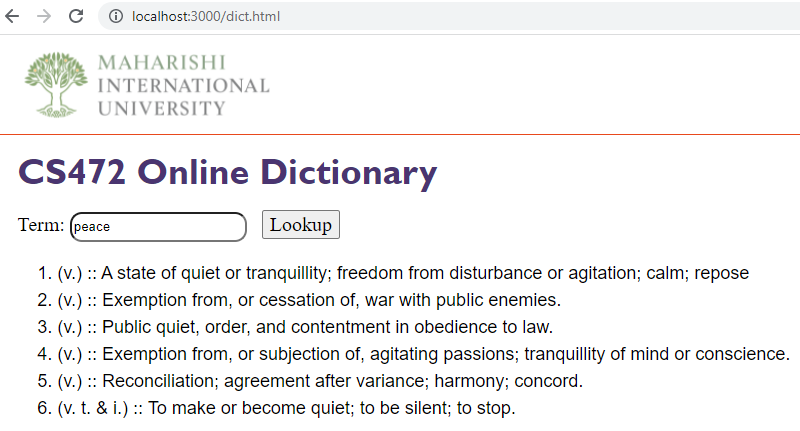
  border-top: #e8491d solid 1px;

}

/ Output – Initial page (Before Search)



// output – Sample outputs for both valid and invalid inputs



If you searched for a nonsense term, you will get the following output.

