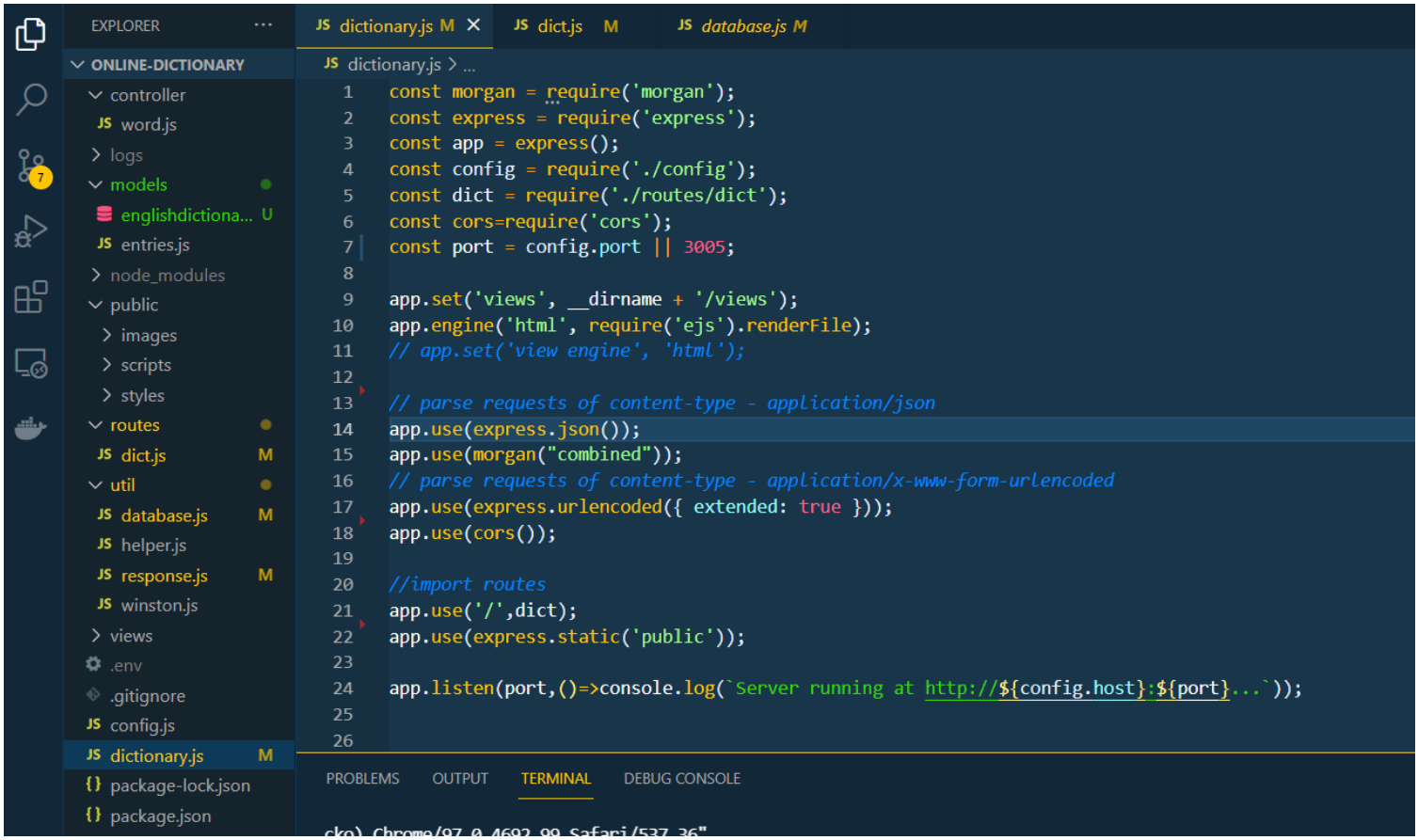
Online Dictionary Project

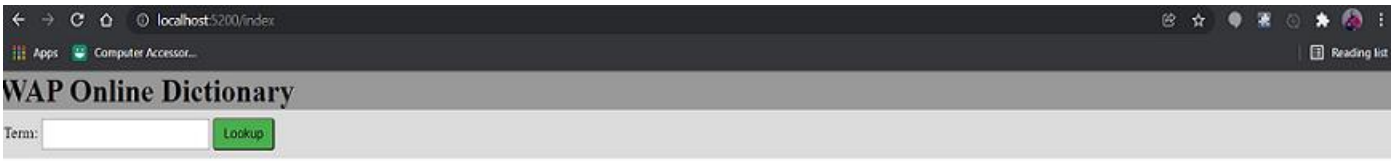
Note: The configurations for the port number, database, localhost can be changed in the .env file of the project. I used npm module called sequelize a promise-based Node.js ORM which works with mysql for the database connection and operations. I also created the schema and class for the database thereby giving the application control over database related changes. The database is automatically created if it has not yet been created and automatically updates tables if there are any changes when the application launches. Note this was done purposely for the production environment can be disabled when using the application in a production environment.

The structure of the project is attached below

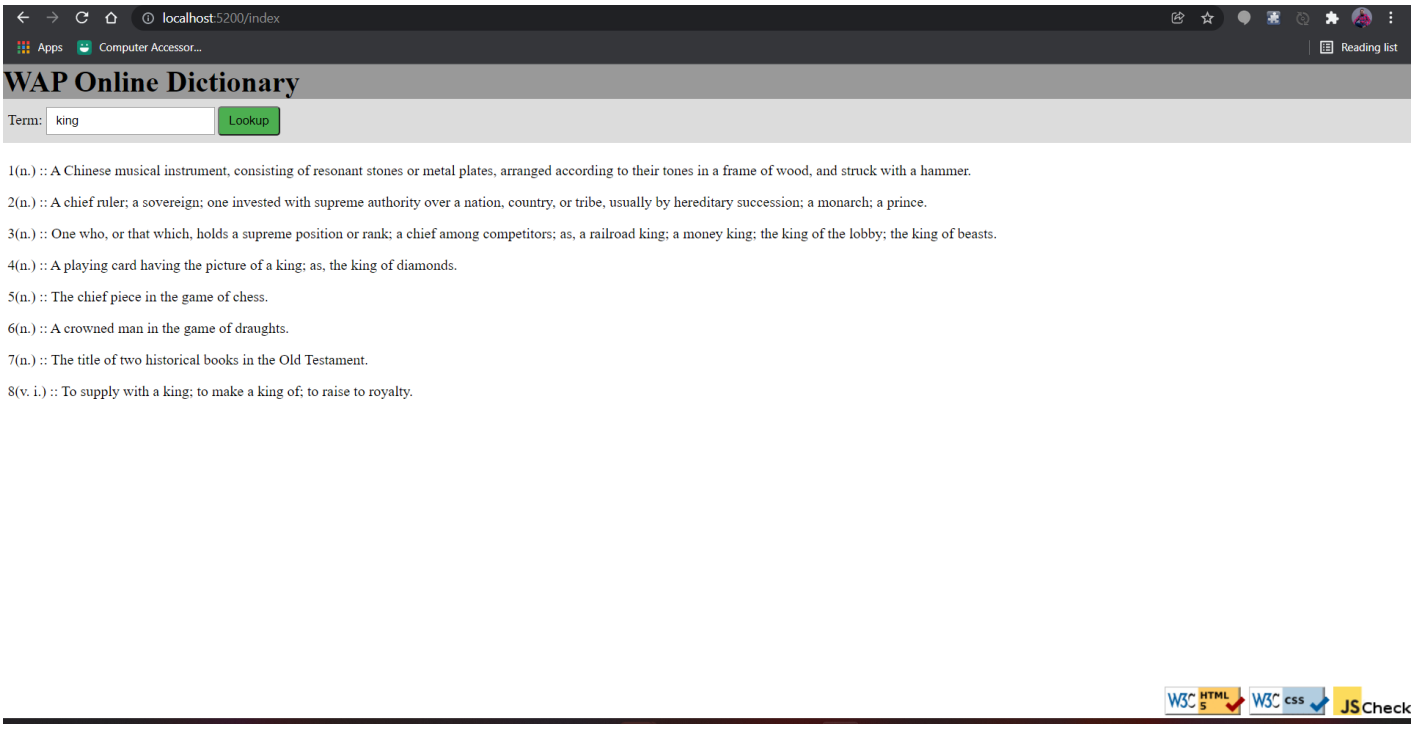


PROJECT SCREENSHOTS

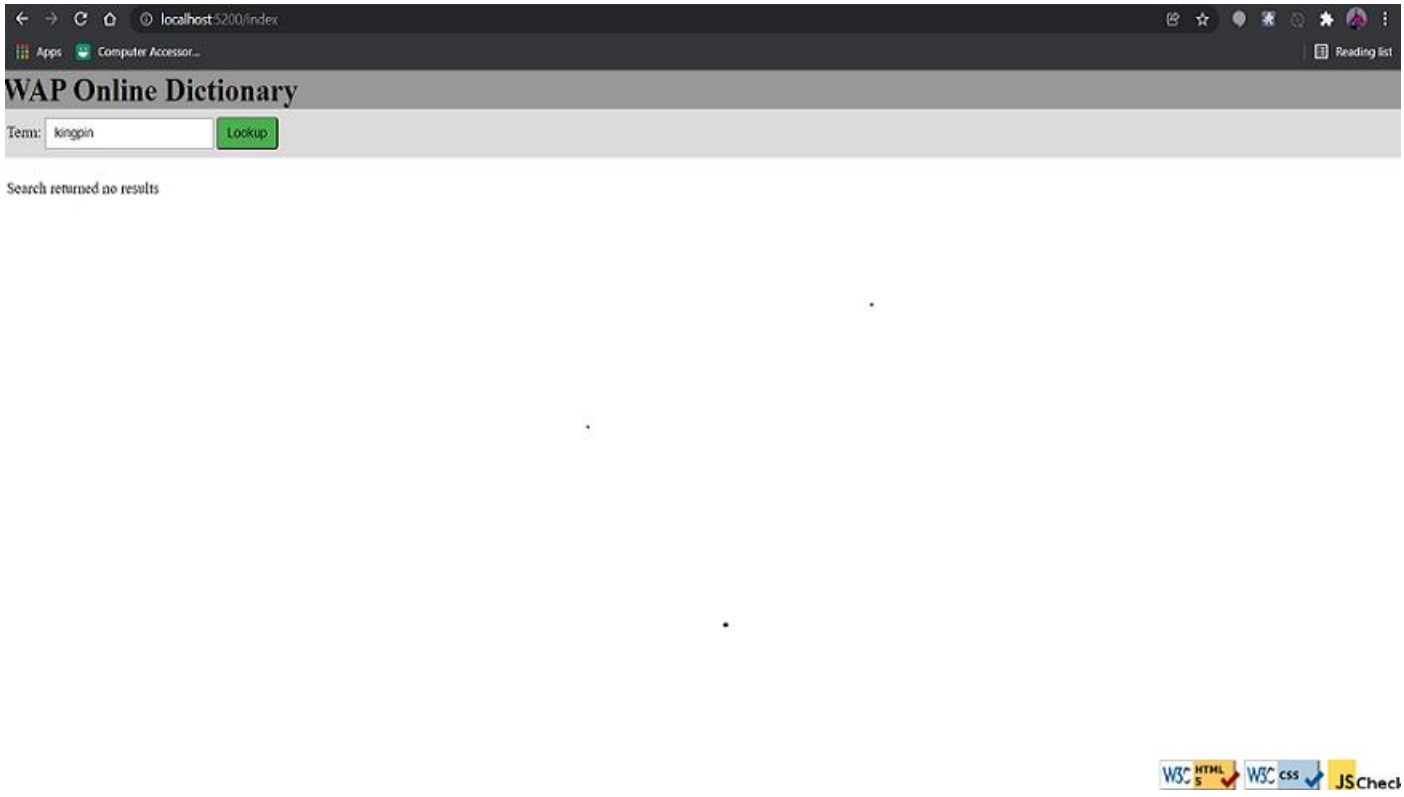
1. Index page



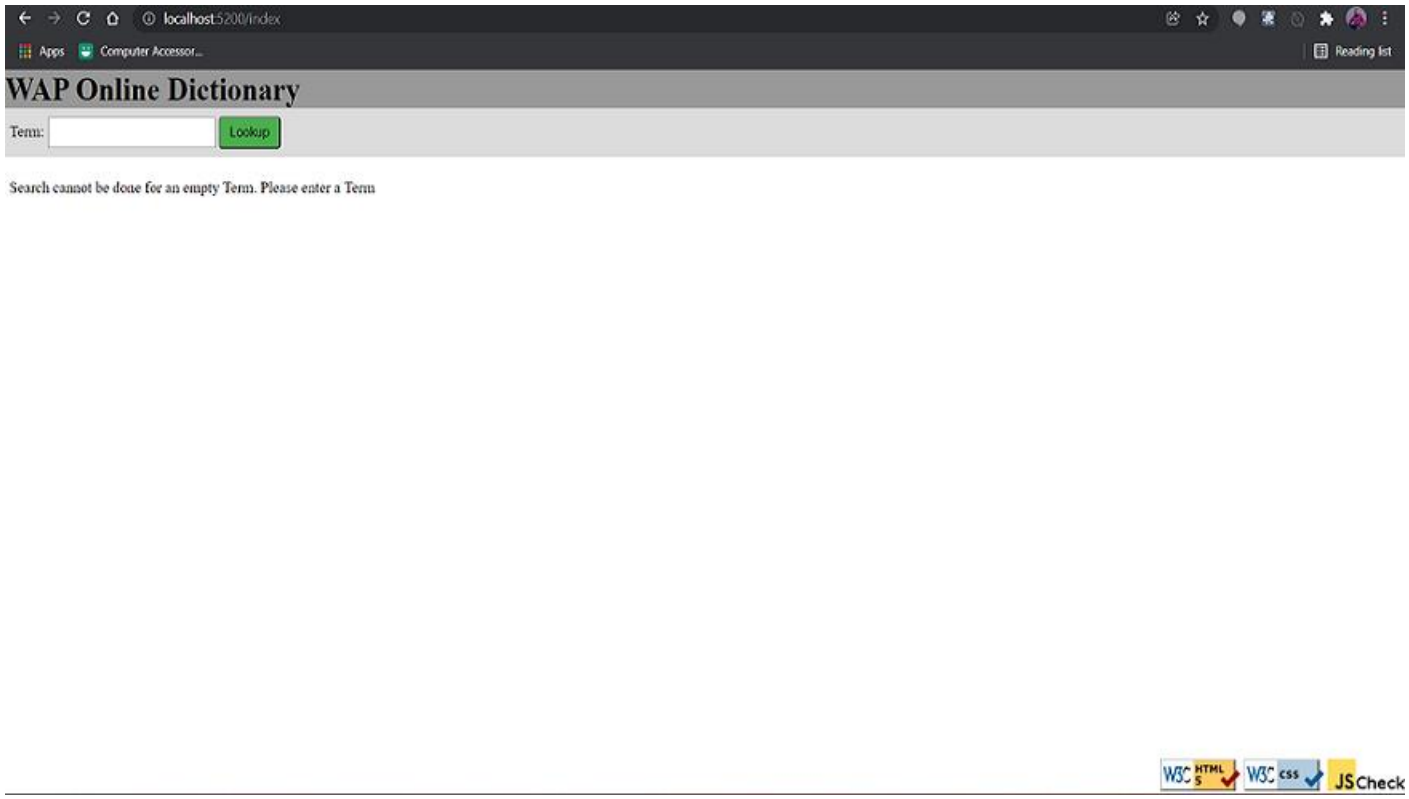
2. Search for a term



3. Search for non-existent term



4. Empty search term



CODE SNIPPETS

1. Database.js

```
const { Sequelize } = require('sequelize');
const config = require('../config');
const Entries = require('../models/entries');

const sequelize = new Sequelize(config.mysql_db.database, config.mysql_db.user,
config.mysql_db.password, {
    host: 'localhost',
    pool: {
        max: 5,
        min: 0,
        acquire: 30000,
        idle: 10000
      },
    dialect:'mysql' /* one of 'mysql' | 'mariadb' | 'postgres' | 'mssql' */
  });

  sequelize
  .authenticate()
  .then(() => {
    console.log('Connection to MYSQL Database established successfully.');
  })
  .catch(err => {
    console.error('Unable to connect to MYSQL database:', err);
  });

  module.exports=sequelize;
```

2. Dictionary.js

```
const morgan = require('morgan');
const express = require('express');
const app = express();
const config = require('./config');
const dict = require('./routes/dict');
const cors=require('cors');
const port = config.port || 3005;

app.set('views', __dirname + '/views');
app.engine('html', require('ejs').renderFile);
// app.set('view engine', 'html');

// parse requests of content-type - application/json
app.use(express.json());
app.use(morgan("combined"));
// parse requests of content-type - application/x-www-form-urlencoded
app.use(express.urlencoded({ extended: false }));
app.use(cors());

//import routes
app.use('/',dict);
app.use(express.static('public'));

app.listen(port,()=>console.log(`Server running at http://${config.host}:${port}...`));
```

3. routes/dict.js

```
const words = require('../controller/word');
const router = require('express').Router();

router.get('/',(req, res)=>{
    res.redirect('/index');
});

router.get('/search',words.searchTerm);
router.get('/index', (req, res)=>{
res.render('dict.html');
});

module.exports=router;
```

4. controller/word.js

```
const Entries = require('../models/entries');
const helper = require('../util/helper');
const respBody = require('../util/response');
const path = require('path');

let wordDefinitions = [];

async function searchTerm (req, res) {
    let word = req.query.word;
    wordDefinitions = [];

    await Entries.findAll({ where: { word: word } })
        .then(data => {
            data.forEach(createResponseObject);
            res.json(respBody.ResponseBody('200', wordDefinitions, ' '));
        })
        .catch(err => {
            res.send(respBody.ResponseBody('400', '', 'Input field validation error' +
helper.parseError(err.message)));
        });
}

async function createResponseObject(value, index) {
    let definition = `${index + 1}(${value.wordtype}) :: ${value.definition}`;
    wordDefinitions.push(definition);
}

module.exports={
    searchTerm,
};
```

5. models/entries.js

```
const { Sequelize, DataTypes } = require('sequelize');
const sequelize = require('../util/database');

const Entries = sequelize.define("entries", {
    word: DataTypes.STRING(25),
    wordtype: DataTypes.STRING(20),
    definition: DataTypes.TEXT
  });

  (async () => {
    // await sequelize.sync({ force: true });
  await  Entries.sync({ alter: true });
    // Code here
  })();

  module.exports=Entries;
```