MINI PROJECT

(2020-21)

Building and Deployment of Web application

Online Examination System

MID-TERM REPORT

Institute of Engineering & Technology

Submitted by: -

Satyam Kumar Jha (181500627)

Shikha Parashar(181500661)

Sachi Tripathi (181500598)

Supervised By: -

Dr. Manoj Varshney
Technical Trainer

Department of Computer Engineering & Applications

GLA University

Mathura- 281406,

INDIA

ABSTRACT

As we all know examination is a test not just for the person who is giving the exam but also, for the person who is managing the examination keeping in mind that everything goes well. Online Examination system removes those little possible discrepancies or faults which takes place during offline examinations. A major advantage of such a system is that student does not have to be physically present at the exam Centre, he can give his exam from anywhere using just a desktop, mobile and an Internet connection. Also, in case of offline exams, there is a delay in calculating the results as it has to be done manually in that case but in online examination system there is a possible way to make sure that the results are declared not much after the student had finished it. It also minimizes the error in calculating the results. This is the basic idea for choosing this project.

Table of Contents

Table of figures

INTRODUCTION

1.1. General Introduction

To build a web-based application for the knowledge and to test the acquired knowledge of the people, where someone can give their exam and get examined of what they have prepared while studying for the examination.

It is an online examination system to test the skills and knowledge acquired by the students during the tenure of their studies. As we are heading towards digital world, this can be a better way of taking examinations as it is more convenient than older ways.

This mainly focuses on building a web based examination portal. It aims to build a system which can offer a convenient way of taking examinations. This has multiple benefits like students don't have to actually visit examination centers or they need not to be physically present over there. They just need a device having a trustworthy internet connection. Also it is very convenient for the officials to conduct exams in online mode as it relieves them of so many different things which they need to manage while conducting examinations in offline mode.

1.1.1. What is a Web application (Web app)

A Web application (Web app) is an application program that is stored on a remote server and delivered over the Internet through a browser interface. Web services are Web apps by definition and many, although not all, websites contain Web apps. According to Web.AppStorm editor Jarel Remick, any website component that performs some function for the user qualifies as a Web app.

Web applications can be designed for a wide variety of uses and can be used by anyone; from an organization to an individual for numerous reasons. Commonly used Web applications can include webmail, online calculators, or e-commerce shops. Some Web apps can be only accessed by a specific browser; however, most are available no matter the browser.

1.1.2. How a web application works

Web applications are usually coded in browser-supported language such as JavaScript and HTML as these languages rely on the browser to render the program executable. Some of the applications are dynamic, requiring server-side processing. Others are completely static with no processing required at the server.

The web application requires a web server to manage requests from the client, an application server to perform the tasks requested, and, sometimes, a database to store the information. Application server technology ranges from ASP.NET, ASP and ColdFusion, to PHP and JSP.

Here's what a typical web application flow looks like:

1. **User** triggers a request to the **web server** over the **Internet**, either through a web browser or the application’s user interface
2. **Web server** forwards this request to the appropriate **web application server.**
3. **Web application server** performs the requested task – such as querying the **database** or processing the data – then generates the results of the requested data
4. **Web application server** sends results to the **web server** with the requested information or processed data
5. **Web server** responds back to the client with the requested information that then appears on the user’s display

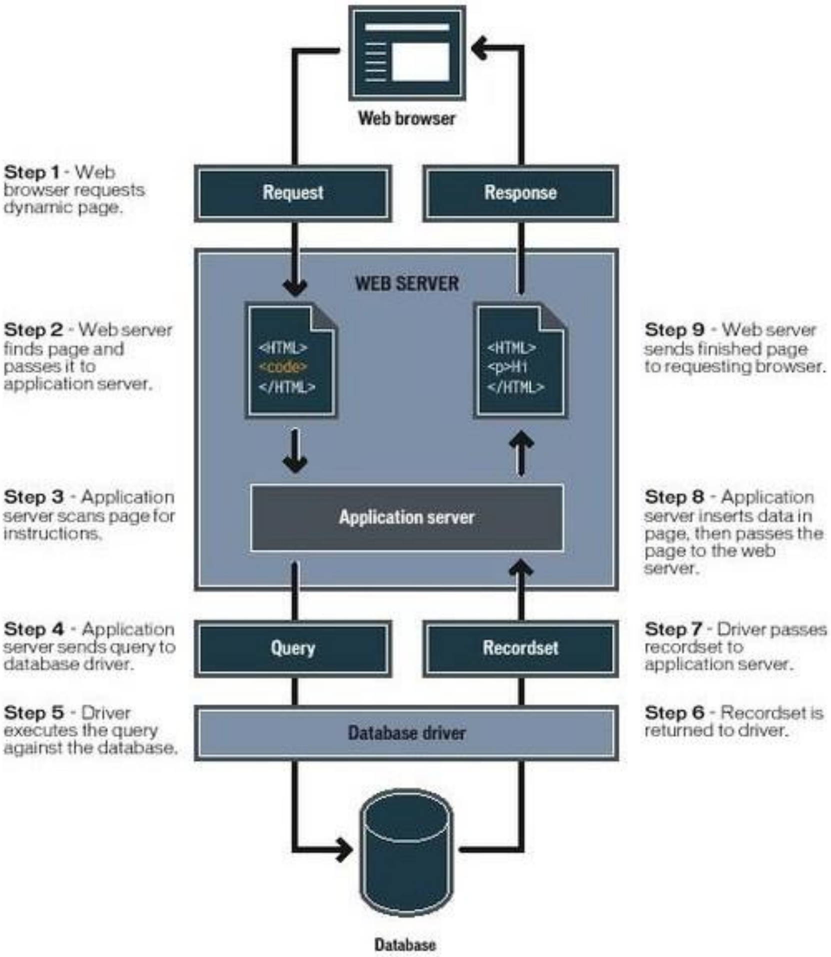

Web browser
Step 1 - Web browser requests dynamic page.
Request
Response
WEB SERVER
Step 2 - Web server finds page and passes it to application server.
<HTML> <code> </HTML>
<HTML> <p>Hi </HTML>
Step 9 - Web server sends finished page to requesting browser.
Step 3 - Application server scans page for instructions.
Application server
Step 8 - Application server inserts data in page, then passes the page to the web server.
Step 4 - Application server sends query to database driver.
Query
Recordset
Step 7 - Driver passes recordset to application server.
Step 5 - Driver executes the query against the database.
Database driver
Step 6 - Recordset is returned to driver.
Database

Figure 1 Web App Working

1.1.3 Introduction to Dependencies

A brief description of each package and the function it will serve

- bcryptjs: used to hash passwords before we store them in our database
- body-parser: used to parse incoming request bodies in a middleware
- concurrently: allows us to run our backend and frontend concurrently and on different ports

- express: sits on top of Node to make the routing, request handling, and responding easier to write
- is-empty: global function that will come in handy when we use validator
- jsonwebtoken: used for authorization
- mongoose: used to interact with MongoDB
- passport: used to authenticate requests, which it does through an extensible set of plugins known as strategies
- passport-jwt: passport strategy for authenticating with a JSON Web Token (JWT); lets you authenticate endpoints using a JWT
- validator: used to validate inputs (e.g. check for valid email format, confirming passwords match)
- Nodemon: Nodemon is a utility that will monitor for any changes in code and automatically restart your server, which is perfect for development.

1.2. Area of Computer Science

Here in this project we have touched the area of Computer Science, Web application in which we have created an online examination system using the JavaScript frameworks, CSS, HTML etc. These are some fields used in this project up till now.

Many of the largest technology companies maintain large scale web applications, providing services such as social media, search, advertising and video and audio streaming.

1.3. Hardware Requirement

The requests of the hardware for the web application are as followed:

1. 64 bits laptop or desktop.

1.4. Software Requirement

To access a web portal of this application, its only need a PC/Laptop/Mobile with an integrated and updated web browser.

Desktop browser: Safari, Chrome, Firefox, Opera, IE9+.

Mobile browsers: Android, Chrome Mobile, iOS Safari.

On the server side: a PC/Web Server which meets these specifications:

1. Window Operating System
2. At least 2 GB RAM and 150 GB Free Space
3. Redis Server Installed
4. Python Compiler Installed

1.4. Dependencies

- Node Js
- React Js
- Java Script
- HTML
- CSS
- Express Js
- Mongo DB

OBJECTIVE

The main objective of this project is to create a web application through which people can give online examinations. It will be a web application developed using Node.JS platform and other supporting technologies.

Implementation Details

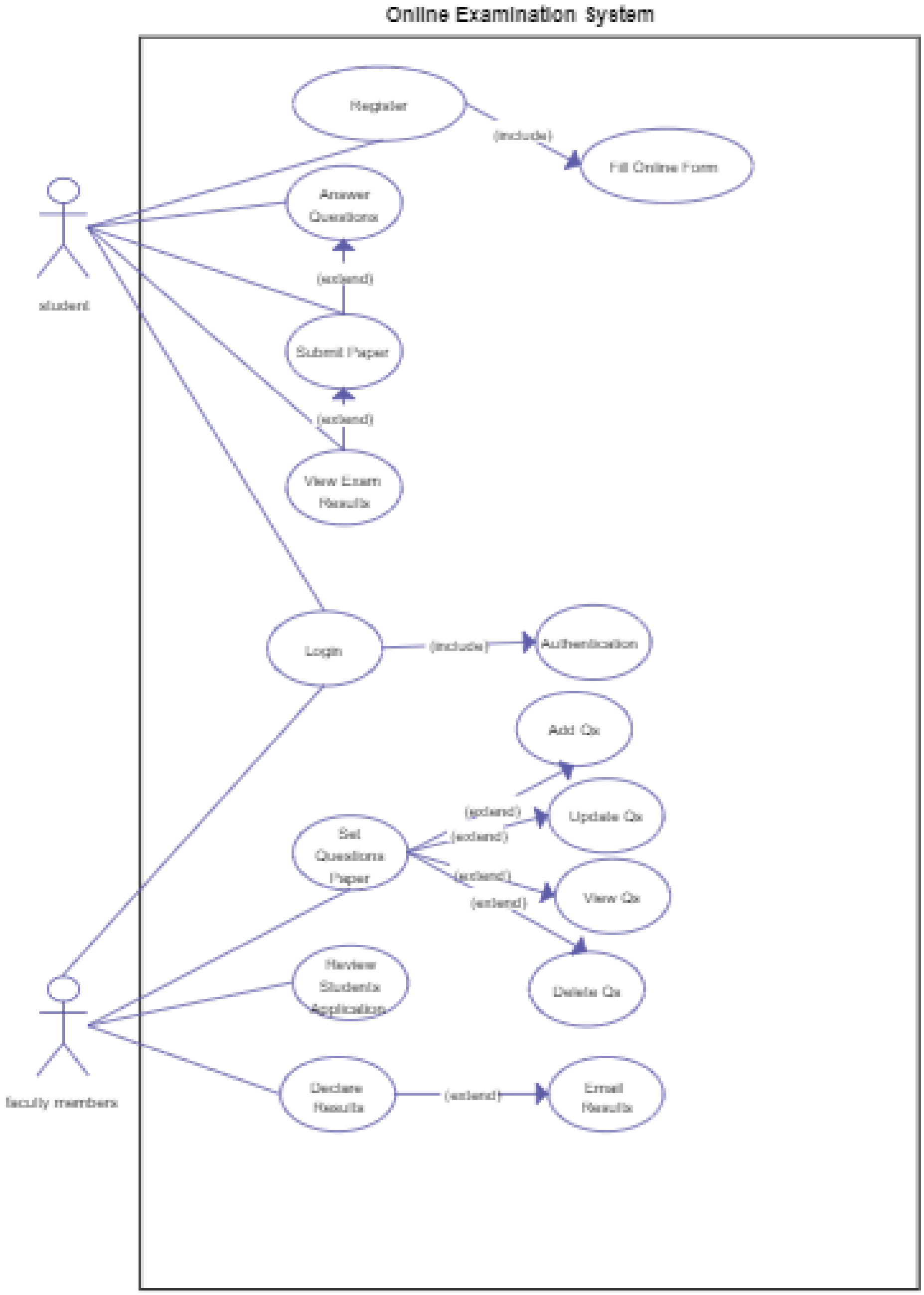


Figure 2 Use Case Diagram

Part 1:

- Install the dependencies.
- Develop functions to connect with MongoDB database.
- Develop function to perform CRUD operations.

Part 2:

- Develop User Interface using React JS.
- Develop necessary scripts for the application.
- Make the application attractive and user friendly.

Part 3:

- Deploy the fully developed web app on Heroku platform.

PROGRESS

Part 1 is completed

Installing the dependencies and developing necessary functions.

Before starting the coding part, we needed to install and configure some softwares and libraries to be used in this project. Those libraries and softwares are as follows:

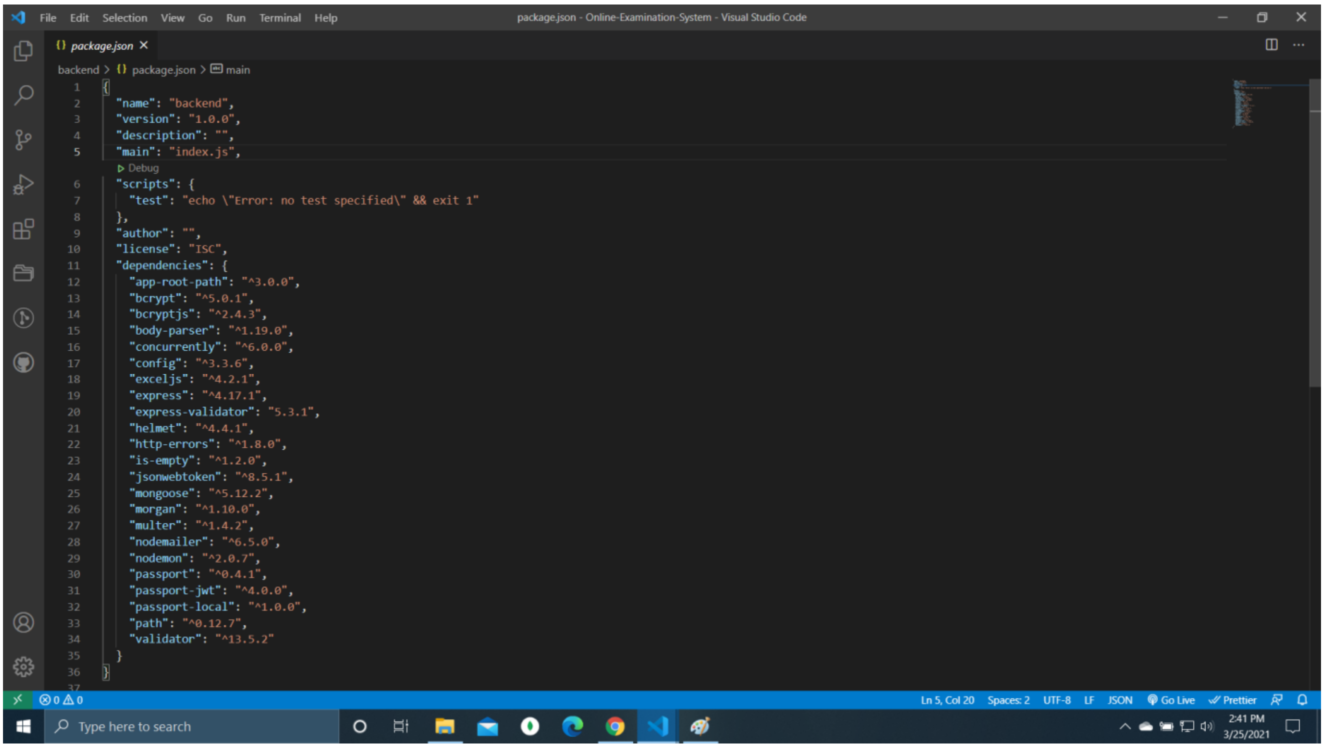
- react@11.2.5
- express@4.17.1
- express-validator@5.3.1
- passport@0.4.1
- passport-jwt@4.0.0
- mongodb
- mongoose@5.12.2
- multer@1.4.2
- nodejs
- nodemon@2.0.7
- bcryptjs@2.4.3

Process we followed for all the installation and configuration is given below:

- Initialize our backend with npm and installed necessary packages .
- Set up a MongoDB database using mLab.

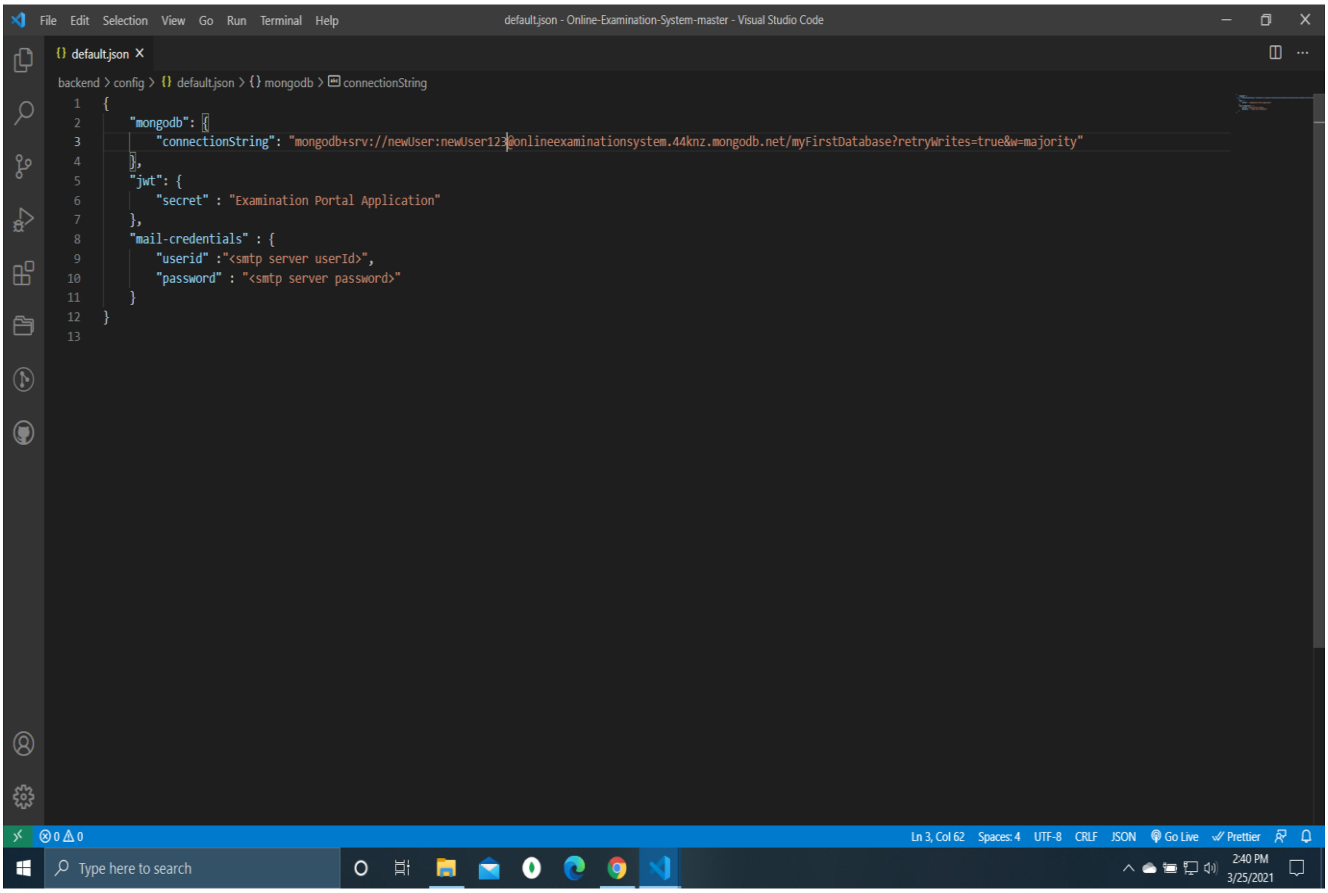
- Set up a server using Node.js and Express.
- Created a database schema to define user for registration and login purposes.
- Set up two API routes, register and login, using passport , jsonwebtoken for authentication and validator for input validation.

SCREENSHOTS



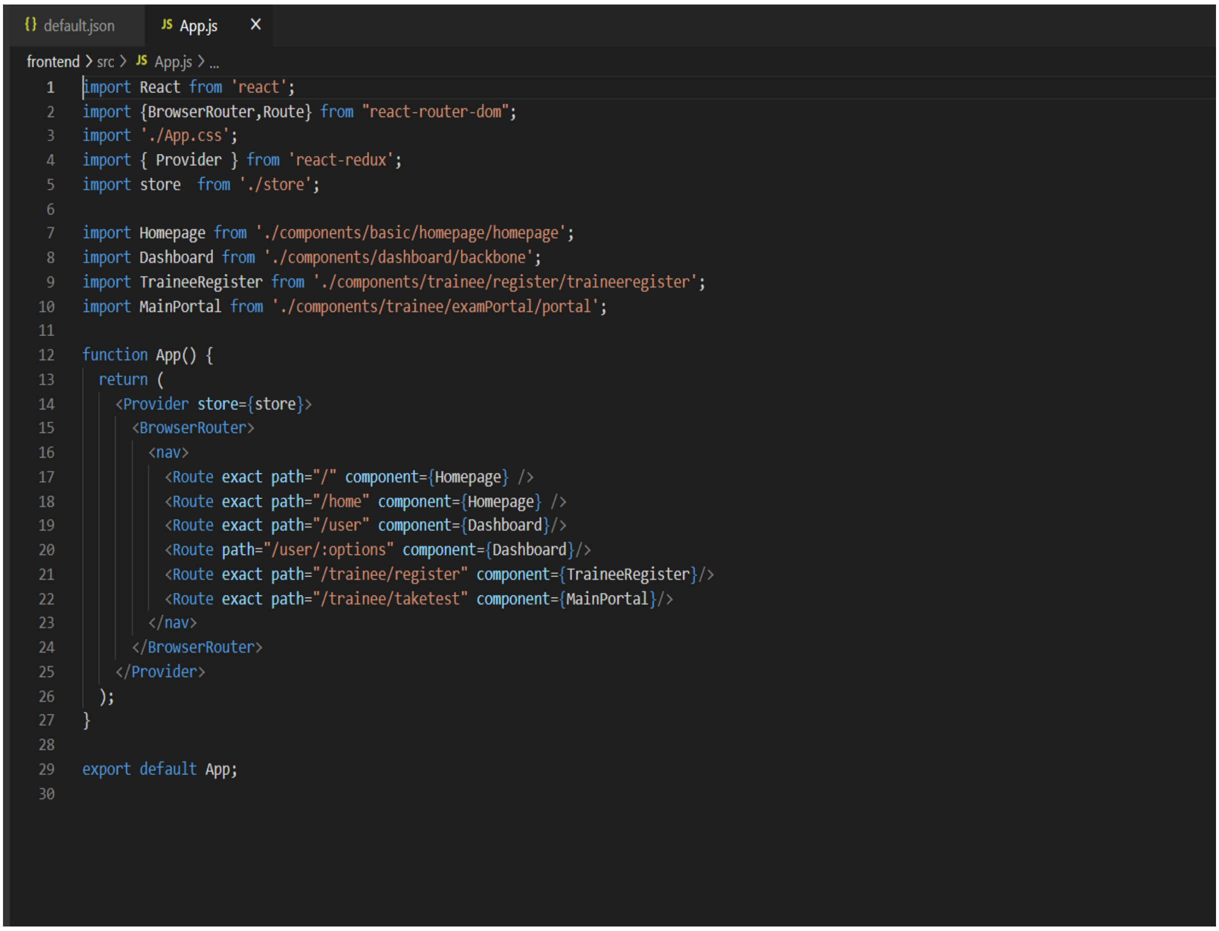
```json
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "app-root-path": "^3.0.0",
    "bcrypt": "^5.0.1",
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.19.0",
    "concurrently": "^6.0.0",
    "config": "^3.3.6",
    "exceljs": "^4.2.1",
    "express": "^4.17.1",
    "express-validator": "5.3.1",
    "helmet": "^4.4.1",
    "http-errors": "^1.8.0",
    "is-empty": "^1.2.0",
    "jsonwebtoken": "^8.5.1",
    "mongoose": "^5.12.2",
    "morgan": "^1.10.0",
    "multer": "^1.4.2",
    "nodemailer": "^6.5.0",
    "nodemon": "^2.0.7",
    "passport": "^0.4.1",
    "passport-jwt": "^4.0.0",
    "passport-local": "^1.0.0",
    "path": "^0.12.7",
    "validator": "^13.5.2"
  }
}
```

Figure 3 package.json for backend


File Edit Selection View Go Run Terminal Help
default.json - Online-Examination-System-master - Visual Studio Code
default.json
backend > config > default.json > mongodb > connectionString
Ln 3, Col 62 Spaces: 4 UTF-8 CRLF JSON Go Live Prettier
Type here to search
2:40 PM
3/25/2021

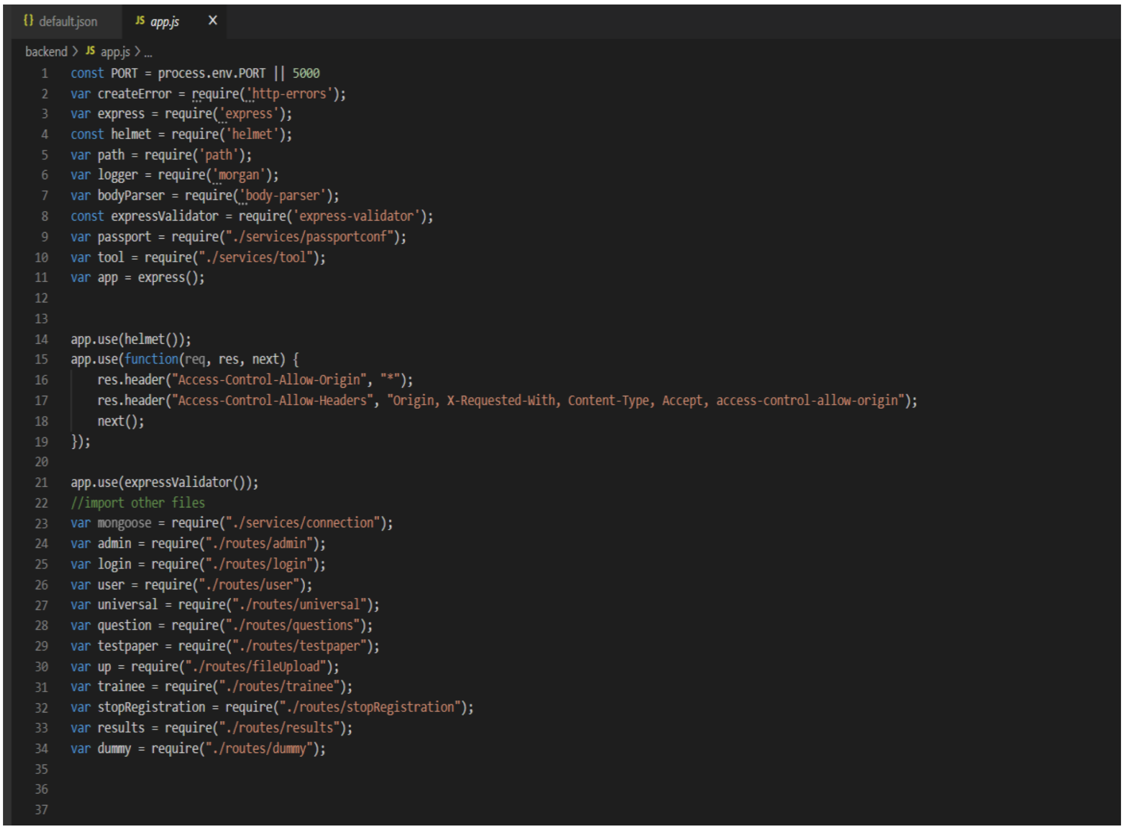
```
{
    "mongodb": {
        "connectionString": "mongodb+srv://newUser:newUser123@onlineexaminationsystem.44knz.mongodb.net/myFirstDatabase?retryWrites=true&w=majority"
    },
    "jwt": {
        "secret" : "Examination Portal Application"
    },
    "mail-credentials" : {
        "userid" :"<smtp server userId>",
        "password" : "<smtp server password>"
    }
}
```

Figure 4 default.json file in config directory



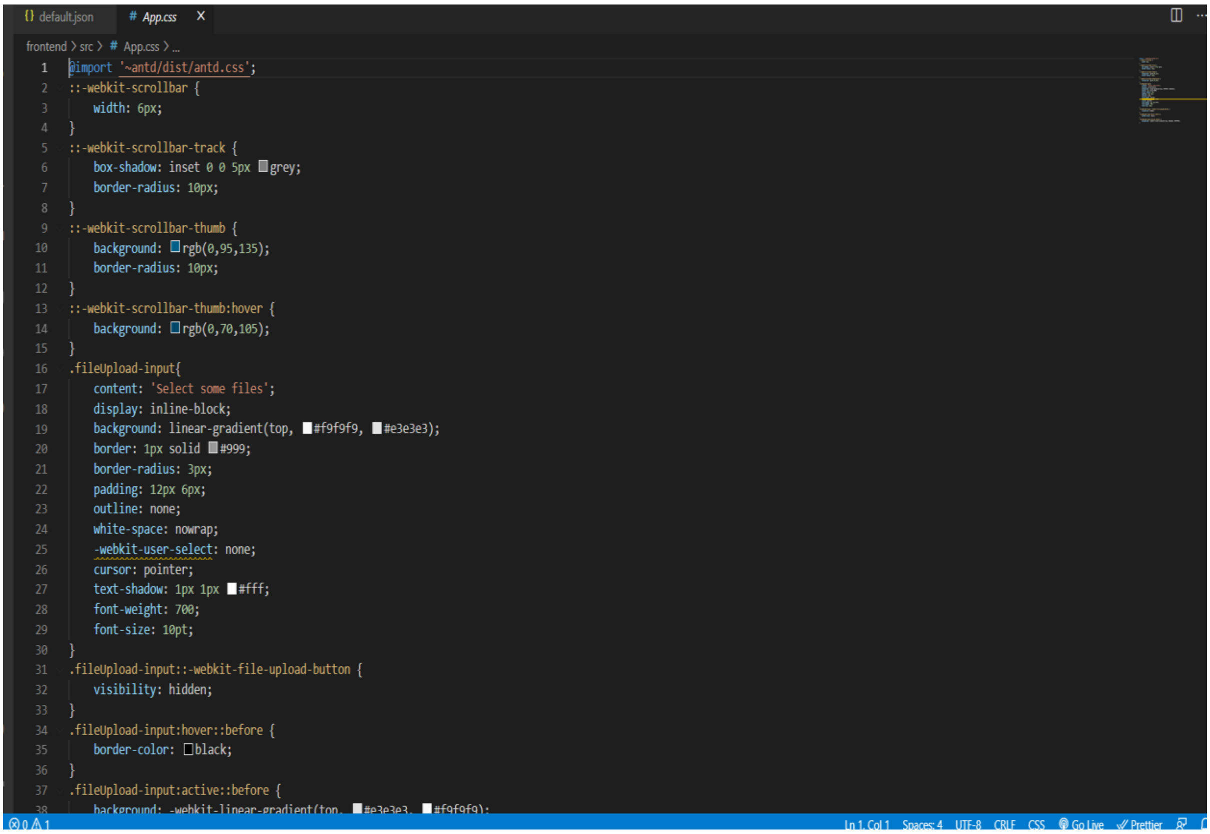
```
import React from 'react';
import {BrowserRouter,Route} from "react-router-dom";
import './App.css';
import { Provider } from 'react-redux';
import store  from './store';

import Homepage from './components/basic/homepage/homepage';
import Dashboard from './components/dashboard/backbone';
import TraineeRegister from './components/trainee/register/traineeregister';
import MainPortal from './components/trainee/examPortal/portal';

function App() {
  return (
    <Provider store={store}>
      <BrowserRouter>
        <nav>
          <Route exact path="/" component={Homepage} />
          <Route exact path="/home" component={Homepage} />
          <Route exact path="/user" component={Dashboard}/>
          <Route path="/user/:options" component={Dashboard}/>
          <Route exact path="/trainee/register" component={TraineeRegister}/>
          <Route exact path="/trainee/taketest" component={MainPortal}/>
        </nav>
      </BrowserRouter>
    </Provider>
  );
}

export default App;
```

Figure 5 App.js for frontend



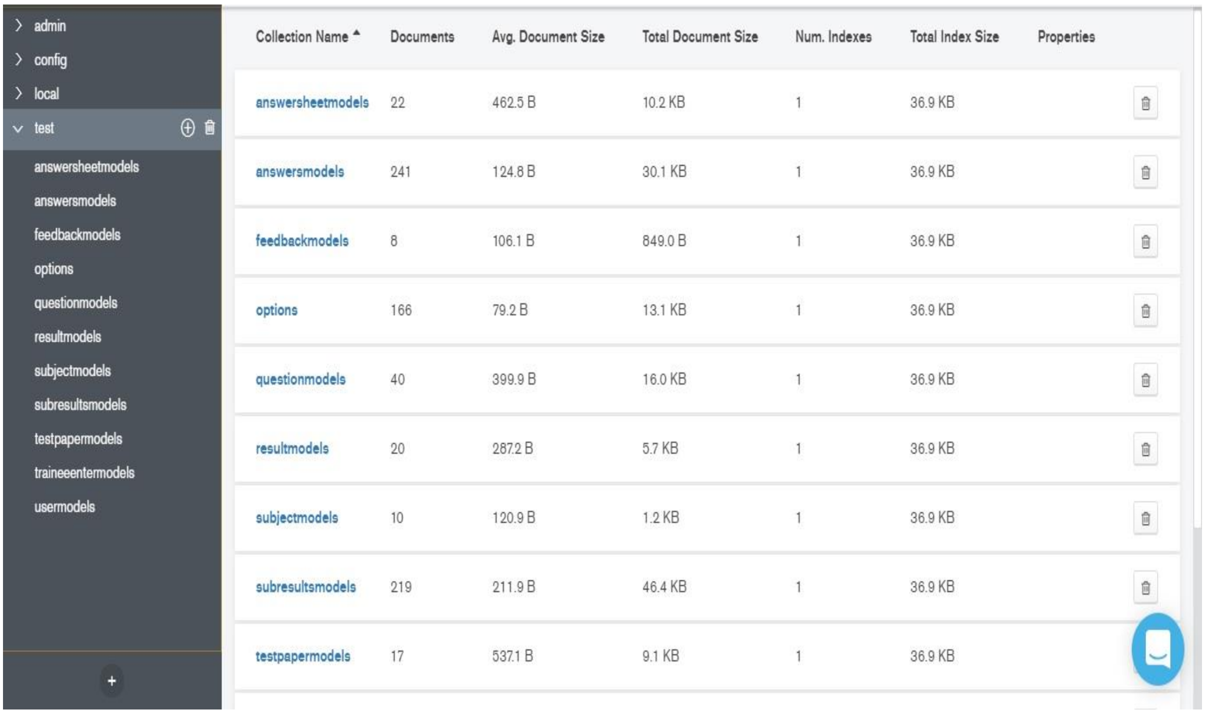
```
const PORT = process.env.PORT || 5000
var createError = require('http-errors');
var express = require('express');
const helmet = require('helmet');
var path = require('path');
var logger = require('morgan');
var bodyParser = require('body-parser');
const expressValidator = require('express-validator');
var passport = require("./services/passportconf");
var tool = require("./services/tool");
var app = express();


app.use(helmet());
app.use(function(req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept, access-control-allow-origin");
    next();
});

app.use(expressValidator());
//import other files
var mongoose = require("./services/connection");
var admin = require("./routes/admin");
var login = require("./routes/login");
var user = require("./routes/user");
var universal = require("./routes/universal");
var question = require("./routes/questions");
var testpaper = require("./routes/testpaper");
var up = require("./routes/fileUpload");
var trainee = require("./routes/trainee");
var stopRegistration = require("./routes/stopRegistration");
var results = require("./routes/results");
var dummy = require("./routes/dummy");
```

Figure 6 app.js in backend directory



```css
@import '~antd/dist/antd.css';
::-webkit-scrollbar {
    width: 6px;
}
::-webkit-scrollbar-track {
    box-shadow: inset 0 0 5px grey;
    border-radius: 10px;
}
::-webkit-scrollbar-thumb {
    background: rgb(0,95,135);
    border-radius: 10px;
}
::-webkit-scrollbar-thumb:hover {
    background: rgb(0,70,105);
}
.fileUpload-input{
    content: 'Select some files';
    display: inline-block;
    background: linear-gradient(top, #f9f9f9, #e3e3e3);
    border: 1px solid #999;
    border-radius: 3px;
    padding: 12px 6px;
    outline: none;
    white-space: nowrap;
    -webkit-user-select: none;
    cursor: pointer;
    text-shadow: 1px 1px #fff;
    font-weight: 700;
    font-size: 10pt;
}
.fileUpload-input::-webkit-file-upload-button {
    visibility: hidden;
}
.fileUpload-input:hover::before {
    border-color: black;
}
.fileUpload-input:active::before {
    background: -webkit-linear-gradient(top, #e3e3e3, #f9f9f9);
```

Figure 7 App.css file



Collection Name ▲	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
answersheetmodels	22	462.5 B	10.2 KB	1	36.9 KB	
answersmodels	241	124.8 B	30.1 KB	1	36.9 KB	
feedbackmodels	8	106.1 B	849.0 B	1	36.9 KB	
options	166	79.2 B	13.1 KB	1	36.9 KB	
questionmodels	40	399.9 B	16.0 KB	1	36.9 KB	
resultmodels	20	287.2 B	5.7 KB	1	36.9 KB	
subjectmodels	10	120.9 B	1.2 KB	1	36.9 KB	
subresultsmodels	219	211.9 B	46.4 KB	1	36.9 KB	
testpapermodels	17	537.1 B	9.1 KB	1	36.9 KB	

Figure 8 mongoDB database Structure

REFERENCES

- https://www.tutorialspoint.com
- https://en.wikipedia.org/wiki/
- https://reactjs.org/tutorial/tutorial.html
- https://www.mongodb.com/
- https://www.npmjs.com/
- https://www.w3schools.com/
- https://nodejs.org/