بنام خداوند بخشنده مهربان

سامانه آزمون آنلاین

سند پیاده سازی داکر پروژه میکروسرویس

استاد محترم: جناب آقای دکتر پرویز رشیدی

دانشجو: الهه سادات خداپرست

نیمسال اول 1402–1403

پروژه ما سامانه آزمون آنلاین می باشد که به صورت یک وب‌اپلیکیشن است که به مدیران و استفاده‌کنندگان اجازه می‌دهد تا آزمون‌ها را به صورت آنلاین برگزار کنند.
پروژه فوق یک وب اپلیکیشن برگزاری آزمون‌های چند گزینه‌ای (MCQ) آنلاین است و از سه نقش اصلی کاربرStudent ، Teacher، Admin پشتیبانی می‌کند. در زیر، توضیحات در مورد تکنولوژی‌ها و ویژگی‌های موردنیاز برای هر نقش ارائه شده است:

فرانت‌اند:(Frontend)

برای بخش کاربری برنامه، م از فریمورک React.js استفاده شده است React.js یکی از پرکاربردترین فریمورک‌های جاوااسکریپت برای ساخت رابط کاربری وب است. این فریمورک اجازه می‌دهد تا به راحتی اجزای قابل استفاده مانند صفحات، فرم‌ها، کنترل‌ها و واکنش‌ها را ساخته و مدیریت کنید.

بک‌اند:(Backend)

برای بخش سمت سرور برنامه، از زبان برنامه‌نویسی Node.js استفاده نموده ایم. Node.js یک محیط اجرایی جاوااسکریپت مبتنی بر سمت سرور است که به ما امکان می‌دهد با استفاده از جاوااسکریپت بر روی سرورها کد بنویسیم. با استفاده از Node.js و فریمورک Express.js ، ما API های سمت سرور را پیاده‌سازی کرده و ارتباط بین کلاینت و سرور را برقرار نمودیم.

- passport

پاسپورت یک ماژول احراز هویت برای Node.js است که امکان ایجاد سیستم‌های ورود و احراز هویت کاربران را فراهم می‌کند.

Node.js یک محیط اجرایی برای اجرای کد جاوااسکریپت در سمت سرور است که اجازه می‌دهد تا برنامه‌های سمت سرور را با استفاده از جاوااسکریپت ایجاد کنند و از آن برای ایجاد برنامه‌های وب پویا استفاده کنند.

- React یک کتابخانه جاوااسکریپت برای ساخت رابط کاربری است که توسط شرکت فیس‌بوک ایجاد شده است. ابزاری برای ساخت وب‌سایت‌ها یا برنامه‌های تحت وب پویا است.
- Express یک چارچوب وب برای Node.js است که از این چارچوب برای فراهم سازی امکاناتی مانند مدیریت درخواست‌ها و پاسخ‌ها، روتینگ و مدیریت واسط‌های برنامه نویسی (API) استفاده شده است.

پایگاه داده:(Database)

برای ذخیره‌سازی و مدیریت اطلاعات مرتبط با کاربران، آزمون‌ها، سوالات و نتایج، از پایگاه داده NoSQL مانند MongoDB استفاده شده . MongoDBیک پایگاه داده بدون ساختار است که بر روی سندها (JSON) کار می‌کند. این نوع پایگاه داده منعطفی است که امکان ذخیره‌سازی و دسترسی به داده‌های پیچیده را فراهم می‌کند.

برای پورتال آزمون آنلاین با ویژگی‌هایی که ذکر شده ما توانستیم یک برنامه کاربردی وب توسعه دهیم که به کاربران دسترسی به این ویژگی‌ها را بدهد. در زیر توضیحاتی درباره هر یک از این ویژگی‌ها آمده است:

ویژگی‌های کاربران دانشجو:

مشاهده جزئیات آزمون: دانشجوان قادر خواهند بود جزئیات آزمون‌ها را مشاهده کنند، مانند نام آزمون، موضوع، تاریخ برگزاری و زمان مجاز برای آزمون.

ثبت‌نام برای آزمون: دانشجوان قادر خواهند بود برای آزمون‌های مورد نظر خود ثبت‌نام کنند.

برگزاری آزمون: دانشجوان توانایی پاسخگویی به سوالات آزمون را خواهند داشت.

بررسی نتایج و پاسخ‌های صحیح: دانشجویان قادر خواهند بود نتایج آزمون خود را مشاهده کنند، از جمله نتیجه کلی، پاسخ‌های صحیح و توضیحات مربوط به سوالات.

ویژگی‌های کاربران معلم:

ایجاد و به‌روزرسانی سوالات و بانک سوالات: معلمان قادر خواهند بود سوالات جدید را ایجاد کنند و سوالات موجود را به‌روزرسانی کنند. همچنین قادر خواهند بود بانک سوالات را مدیریت کنند.

ایجاد و مشاهده آزمون: معلمان قادر خواهند بود آزمون‌های جدید را ایجاد کنند و جزئیات آزمون‌ها را مشاهده کنند، از جمله لیست سوالات مربوطه و تنظیمات آزمون (مانند زمان مجاز و تاریخ برگزاری).

ویژگی‌های کاربران مدیر:

ایجاد و مدیریت کاربران معلم: مدیران قادر خواهند بود کاربران معلم را ایجاد و مدیریت کنند، از جمله ایجاد حساب کاربری، تنظیمات دسترسی و تغییر رمز عبور.

ایجاد و مدیریت موضوعات: مدیران قادر خواهند بود موضوعات مختلف را ایجاد و مدیریت کنند، برای دسته‌بندی سوالات و آزمون‌ها.

با پیاده‌سازی این ویژگی‌ها در پورتال آزمون آنلاین، کاربران با نقش‌های مختلف (دانشجو، معلم و مدیر) قادر خواهند بود از قابلیت‌های مورد نیاز خود بهره‌برد. این پورتال به ما امکان می‌دهد آزمون‌های آنلاین را برای دانشجوان برگزار کنید، معلمان بتوانند سوالات را ایجاد و مدیریت کنند، و مدیران قادر خواهند بود کاربران و موضوعات را مدیریت کنند.

توسعه یک پورتال آزمون آنلاین یک پروژه گسترده بوده و به صورت پیاده سازی میکروسرویس بیس انجام شده است. پروژه ما دارای ساختار چند لایه (multi-layered) یا معماری مایکروفرانت‌اند، به عنوان یک سامانه متشکل از ماژول‌های جداگانه که هرکدام یک قسمت از ویژگی‌ها را پوشش می‌دهد، طراحی شده است. در این ساختار، هر ماژول یا لایه مستقل از دیگر لایه‌ها به‌عنوان یک میکروسرویس عمل می‌کند.

در ساختار چند لایه یا معماری مایکروفرنت‌اند، فرانت‌اند نیز به صورت جداگانه می‌تواند تقسیم شود. این معماری به تفکیک وظایف مختلف فرانت‌اند را ترویج می‌کند. به عبارت دیگر، هر قسمت از ویژگی‌ها یا صفحات خاص به عنوان یک فرانت‌اند مجزا از سایر فرانت‌اندها عمل می‌کند.

ما دو فرانت‌اند با نام‌های frontend و user-frontend در پروژه داریم، این نشان‌دهنده دو بخش مختلف از پروژه باشد. پورتال فرانت‌اند یوزر ویژگی‌ها یا صفحات مربوط به یک ناحیه خاص از سامانه را در بر دارد. این تقسیم بندی می‌تواند بر اساس نیازهای کاربران، نقش‌ها، یا قابلیت‌های خاص سیستم انجام شود.

به‌عنوان مثال، user-frontend می‌تواند شامل فرانت‌اند مرتبط با ویژگی‌ها یا صفحاتی برای کاربران عادی باشد، در حالی که frontend می‌تواند برای مدیران یا افراد با دسترسی‌های خاص به منظور مدیریت و پیکربندی سیستم است.

با استفاده از این ساختار، توسعه و مدیریت فرانت‌اندها به صورت مستقل انجام شده و هر قسمت می‌تواند بدون تأثیر بر سایر قسمت‌ها ارتقاء یابد. همچنین، این ساختار مجازی‌سازی توسعه (micro frontends) نامیده می‌شود.

همچنین در این پروژه، به منظور اجرای بخش سرور (backend) و ارتباط با پایگاه داده MongoDB و همچنین برای پیاده‌سازی JSON Web Token (JWT) برای امنیت و احراز هویت، از متغیرهای محیطی (Environment Variables) استفاده می‌شود. این متغیرها در یک فایل تنظیمات (config.json) قرار داده شده‌اند. در زیر توضیحاتی در مورد هرکدام از این متغیرها آورده شده است:

1: mongodb.connectionString .

این متغیر محیطی مشخص می‌کند که چگونه برنامه به پایگاه داده MongoDB متصل شود. معمولاً این متغیر حاوی رشته‌ای است که شامل اطلاعات مربوط به میزبان(host) ، پورت، نام پایگاه داده و اطلاعات احراز هویت به منظور اتصال به MongoDB است.

2: jwt.secret .

این متغیر محیطی مشخص می‌کند که چه کلیدی برای امضای (signing) توکن‌های JWT استفاده شود. توکن JWT برای احراز هویت و تأیید صحت کاربران استفاده می‌شود. کلید محرمانه (secret key) باید به‌طور محرمانه نگهداری شود تا افراد غیرمجاز به آن دسترسی نداشته باشند.

به طور خلاصه، JSON Web Token (JWT) یک استاندارد باز و امن برای ارسال اطلاعات در قالب یک توکن است. این توکن بصورت رشته‌ای ساختار یافته است و شامل سه بخش اصلی است: سربرگ(Header) ، بدنه (Payload) و امضا.(Signature)

Backend: برنامه‌ای است که در سمت سرور اجرا می‌شود و وظیفه پردازش داده‌ها، ارسال درخواست‌ها به پایگاه داده و ارسال پاسخ به فرانت‌اند را دارد. کد برنامه شامل موارد زیر است:

- Config .تنظیمات مربوط به اتصال به پایگاه داده و سایر تنظیمات سرور.
- Models .تعریف مدل‌های داده‌ای که برای ذخیره اطلاعات در پایگاه داده استفاده می‌شود.
- Public .فایل‌ها و منابع عمومی که بستری برای ارتباط با سرور ایجاد می‌کنند.
- Routes .تعریف مسیرها و درخواست‌های مربوط به هر قسمت از برنامه.
- Schemas .تعریف ساختار داده‌ها و اطلاعات مورد نیاز برای ذخیره در پایگاه داده.
- Service .لایه‌ی سرویس‌دهی که وظیفه پردازش درخواست‌ها و ارسال پاسخ‌ها را دارد.
- Test: تست‌های واحد و اجزا برای اطمینان از صحت عملکرد بخش‌های مختلف بک‌اند.

داکر فایل backend

این فایل Docker به ما کمک می‌کند تا یک محیط اجرایی برای برنامه‌ی بک‌اند خود ایجاد کنیم. این فایل از تصویر node:13-alpine به عنوان پایه استفاده می‌کند و سپس کد برنامه را درون محیط اجرایی کپی می‌کند. سپس با استفاده از دستورات RUN، npm و CMD، محیط اجرایی را پیکربندی کرده و برنامه را اجرا می‌کند.

Frontend: برنامه‌ای است که در سمت مرورگر کاربر اجرا می‌شود و وظیفه نمایش رابط کاربری، ارسال درخواست‌ها به بک‌اند و نمایش داده‌های دریافتی را دارد.

بخش frontend شامل دو بخش اصلی است public و .src بخش public شامل فایل‌های عمومی مانند تصاویر و فایل‌های استاتیک است، و بخش src شامل کدهای منبع برنامه و فایل‌های اصلی است.

. public:1

دایرکتوری public در پروژه شامل فایل‌ها و منابعی است که مستقیماً توسط مرورگر اجرا می‌شوند و در دسترس عموم قرار می‌گیرند. این فایل‌ها عمدتاً برای استاتیک‌سازی مثل تصاویر، فونت‌ها، favicon و ... مورد استفاده قرار می‌گیرند.

برخی از فایل‌ها و موارد در دایرکتوری public عبارتند از:

index.html: این فایل معمولاً به عنوان نقطه شروع اصلی برنامه React استفاده می‌شود.

favicon.ico: آیکون سایت که در تب مرورگر نمایش داده می‌شود.

. src:2

دایرکتوری src محل قرار گرفتن کد منبع (source code) اصلی برنامه React است. این دایرکتوری به‌طور معمول حاوی فایل‌های جاوااسکریپت JSX یا JS ، کدهای CSS یا فایل‌هایSass ، تصاویر و سایر فایل‌های مورد نیاز برنامه می‌باشد.

فایل‌ها و موارد دایرکتوری src عبارتند از:

index.jsیا index.jsx فایل اصلی که React را به DOM متصل می‌کند.

App.jsیا App.jsx کامپوننت اصلی برنامه که به عنوان نقطه شروع برنامه React عمل می‌کند.

Componentsیک دایرکتوری مخصوص نگهداری کامپوننت‌های.React

Styles یک دایرکتوری مخصوص قرار دادن فایل‌های CSS برای استایل‌دهی به کامپوننت‌ها.

داکر فایل frontend

در داکر فایل برای ساخت تصویر Docker در اجرای برنامه فرانت‌اند (Frontend) از زبان Node.js استفاده نمودیم. در زیر توضیحاتی در مورد هر بخش از این Dockerfile آورده شده است:

. FROM node:13-alpine:1

این دستور تصمیم می‌گیرد که تصویر ابتدایی از Node.js با نسخه 13 و بر اساس سیستم عامل Alpine را به عنوان پایه برای تصویر ساخته شده انتخاب کند.

نسخه Alpine از سیستم عامل لینوکس به عنوان یک سیستم عامل سبک و کوچک شناخته می‌شود.

. WORKDIR /usr/frontend:2

تعیین محل کاری (working directory) به/usr/frontend ، جایی که تمام فایل‌ها و فرآیندهای بعدی اجرا خواهند شد.

. COPY . .:3

این دستور تمام فایل‌ها و پوشه‌های فرانت‌اند را از مسیر فعلی (context) محل ساخت Docker که به عنوان مسیر context در فایل-docker compose مشخص شده است به مسیر /usr/frontend در کانتینر کپی می‌کند.

. RUN rm -rf node_modules/ package-lock.json:4

این دستور node_modules و package-lock.json را از مسیر فعلی حذف می‌کند. این اقدام معمولاً برای جلوگیری از اشکالات در نصب وابستگی‌ها از فایل package-lock.json و حذف node_modules قبلی استفاده می‌شود.

. RUN npm install:5

نصب وابستگی‌های پروژه با استفاده از npm.

. RUN npm run build:6

اجرای دستور npm run build که عملیات ساخت (build) پروژه را انجام می‌دهد. مراحلی مانند ترجمه و ادغام فایل‌های جاوااسکریپت، استفاده از مدیر بسته‌ها (package manager) مانند webpack یا react-scripts و غیره را شامل می شود.

. CMD ["sh", "-c", "npm start"]:7

تعیین دستوری که هنگام اجرای کانتینر اجرا خواهد شد. در اینجا، این دستور به npm start اشاره دارد که معمولاً برای اجرای سرور توسعه در محیط توسعه محلی استفاده می‌شود. از شل sh به عنوان محیط اجرایی برای اجرای npm start استفاده شده است.

این Dockerfile به‌عنوان دستورات موردنیاز برای ساخت تصویر کانتینر فرانت‌اند ما استفاده می‌شود.

User Portal Frontend: برنامه‌ای است که برای کاربران نهایی طراحی شده است و وظیفه نمایش اطلاعات شخصی، امکانات و قابلیت‌های مربوط به حساب کاربری را دارد.

همانند فرانت اند شامل src و public و داکر فایل می باشد که مطابق فرمت قابل نمایش کاربری است.

Docker Compose

عناصر اصلی:

فایل docker-compose.yml برای تعریف و مدیریت چندین سرویس در یک برنامه و ایجاد ارکستریشن استفاده می‌شود. در اینجا سرویسهایی بدین شرح تعریف شده‌اند backend ، mongodb ، frontend، mongo express و usert frontend.

سرویس backend از تصویر backend-app:1.0 استفاده می‌کند و در پورت 5000 اجرا می‌شود. همچنین به سرویس mongodb متصل شده و متغیر محیطی NODE_ENV را به docker تنظیم می‌کند.

سرویس mongodb از تصویر mongo استفاده می‌کند و در پورت 27018 اجرا می‌شود. همچنین دارای یک نام کاربری و رمز عبور برای دسترسی به پایگاه داده است. این سرویس همچنین دارای یک حالت سلامتی (healthcheck) است که به صورت دوره‌ای وضعیت پایگاه داده را بررسی می‌کند.

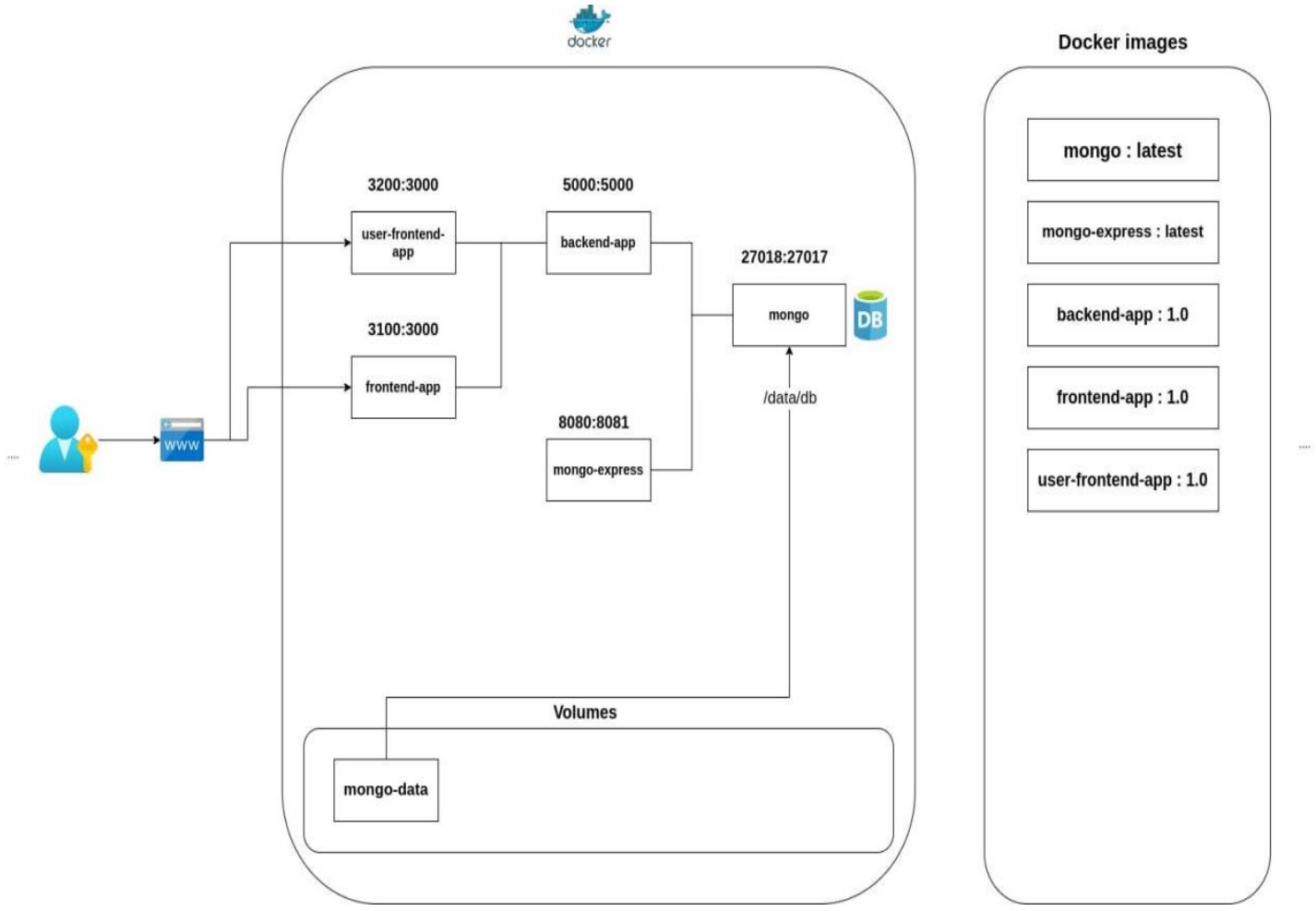
همچنین از volumes برای ذخیره داده‌های پایگاه داده استفاده شده است.

سرویس mongo-express برای مدیریت و مشاهده داده‌های پایگاه داده MongoDB استفاده می‌شود. این سرویس از تصویر mongo-express استفاده می‌کند و در پورت 8080 اجرا می‌شود. همچنین به سرویس mongodb متصل شده و متغیرهای محیطی برای دسترسی به پایگاه داده MongoDB را تنظیم می‌کند.

سرویس frontend از تصویر frontend-app:1.0 استفاده می‌کند و در پورت 3100 اجرا می‌شود. این سرویس نیاز به سرویس backend دارد و از آن وابستگی دارد.

سرویس user-frontend نیز از تصویر user-frontend-app:1.0 استفاده می‌کند و در پورت 3200 اجرا می‌شود. این سرویس نیاز به سرویس‌های backend و frontend دارد و به آن‌ها وابستگی دارد.

همچنین از volumes برای ذخیره داده‌های پایگاه داده MongoDB استفاده شده است.

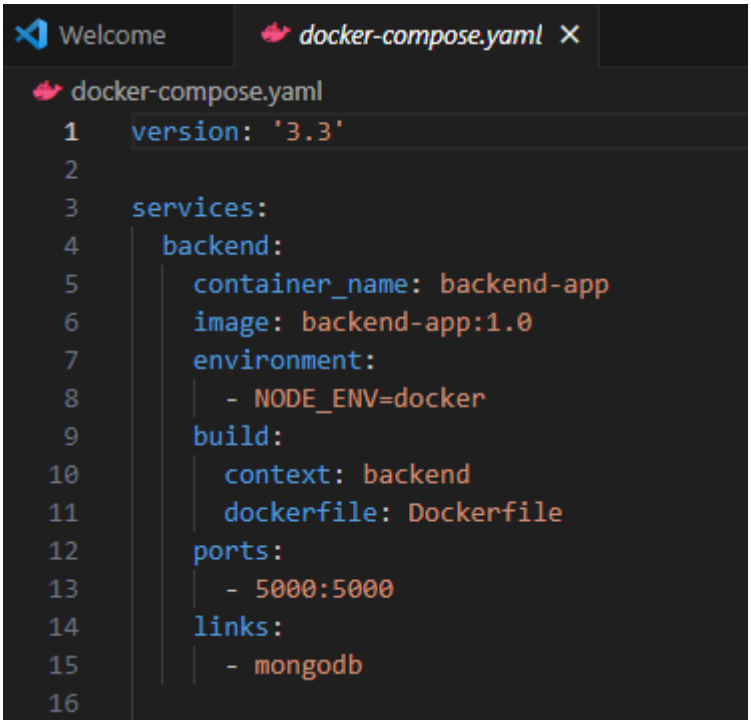


شرح کد داکر کامپوز:

backend

تنظیماتی که برای سرویس بک‌اند در محیط داکرداریم، در این تنظیمات، یک کانتینر با نام backend-app ایجاد می‌شود و از تصویر backend-app:1.0 استفاده می‌کند. متغیر محیطی NODE_ENV به مقدار docker تنظیم شده است و برای ساخت کانتینر از

Dockerfile در مسیر backend استفاده می‌شود. همچنین، پورت 5000 کانتینر به پورت 5000 میزبان متصل شده و به سرویس mongodb لینک شده است.



container_name نام کانتینر را به "backend-app" تعیین می‌کند .

image تصویر Docker مورد استفاده برای سرویس را مشخص کرده؛ در اینجا."backend-app:1.0"

environment متغیر محیطی NODE_ENV را به "docker" تنظیم می‌کند.

build محل و Dockerfile برای ساخت تصویر را مشخص می‌کند. محل ساخت backend است و Dockerfile با نام."Dockerfile"

ports پورت 5000 از میزبان را به پورت 5000 در کانتینر می‌نگارد.

links اتصال به یک سرویس دیگر به نام mongodb ایجاد می‌کند.

MongoDB

این سرویس مربوط به دیتابیس MongoDB است که از تصویر mongo استفاده می‌کند و پورت 27017 داخل کانتینر به پورت 27018 میزبان متصل شده است. همچنین، نام کاربری و رمز عبور برای دیتابیس تنظیم شده و یک Volume برای ذخیره داده‌ها ایجاد شده است. سرویس دوم مربوط به محیط مدیریت دیتابیس MongoDB است که از تصویر mongo-express استفاده می‌کند و به پورت 8081 داخل کانتینر به پورت 8080 میزبان متصل شده است. این سرویس نیز به سرویس mongodb وابسته است تا پس از آماده‌سازی دیتابیس، اجرا شود.

mongodb

image: mongo مشخص می‌کند که از تصویر MongoDB رسمی استفاده می‌شود.

ports: - 27018:27017 پورت 27017 در داخل کانتینر MongoDB را به پورت 27018 در میزبان نگاشت می‌دهد.

environment متغیرهای محیطی برای تنظیم نام کاربری و رمز عبور ریشه MongoDB را مشخص می‌کند.

volumes یک والیوم به نام mongo-data به /data/db درون کانتینر MongoDB اختصاص می‌دهد که برای ذخیره داده‌های پایدار استفاده می‌شود.

healthcheck یک تست سلامتی برای MongoDB اجرا می‌کند تا اطمینان حاصل شود که سرویس در حال اجرا و سالم است.

```yaml
docker-compose.yaml
      - 5000:5000
    links:
      - mongodb


  mongodb:
    image: mongo
    ports:
      - 27018:27017
    environment:
      - MONGO_INITDB_ROOT_USERNAME=admin
      - MONGO_INITDB_ROOT_PASSWORD=password
    volumes:
      - mongo-data:/data/db
    healthcheck:
      test: echo 'db.runCommand("ping").ok' | mongo mongo:27017/test --
      interval: 10s
      timeout: 10s
      retries: 5
      start_period: 40s
```

mongo-express

image: mongo-express مشخص می‌کند که از تصویر mongo-expressاستفاده می‌شود.

restart: always تنظیم می‌کند که در صورتی که سرویس خاموش شود، همواره مجدداً راه اندازی شود.

ports: - 8080:8081 پورت 8081 در داخل کانتینر mongo-expressرا به پورت 8080 در میزبان نگاشت می‌دهد.

environment متغیرهای محیطی برای تنظیم نام کاربری، رمز عبور ریشه MongoDB و آدرس سرور MongoDB را مشخص می‌کند.

depends_on: - "mongodb"تعیین می‌کند که سرویس mongo-express باید پس از راه اندازی سرویس MongoDB شروع به کار کند.

Frontend

```yaml
  mongo-express:
    image: mongo-express
    restart: always
    ports:
      - 8080:8081
    environment:
      - ME_CONFIG_MONGODB_ADMINUSERNAME=admin
      - ME_CONFIG_MONGODB_ADMINPASSWORD=password
      - ME_CONFIG_MONGODB_SERVER=mongodb
    depends_on:
      - "mongodb"
```

این تنظیمات برای ایجاد یک سرویس فرانت‌اند در یک محیط داکر است. در این تنظیمات، یک کانتینر با نام frontend-app ایجاد شده و از تصویر frontend-app:1.0 استفاده می‌کند. متغیرهای محیطی NODE_ENV و REACT_APP_API_BASE_URLتنظیم شده‌اند و برای ساخت کانتینر از Dockerfile در مسیر frontend استفاده می‌شود. همچنین، پورت 3000 داخل کانتینر به پورت 3100 میزبان متصل شده و به سرویس بک‌اند وابسته است تا پس از آماده‌سازی، اجرا شود.

container_name: frontend-app: نام کانتینر فرانت‌اند را به "frontend-app" تعیین می‌کند.

image: frontend-app:1.0: تصویر Docker مورد استفاده برای سرویس را مشخص می‌کند.

environment:متغیرهای محیطی را برای تنظیم محیط اجرای فرانت‌اند مشخص می‌کند. در اینجا NODE_ENV به "docker" و REACT_APP_API_BASE_URLبه "http://backend:5000" تنظیم شده‌اند.

build: تنظیمات مربوط به ساخت تصویر را مشخص می‌کند .context به معنای مسیر محل فایل‌های ساخت است و dockerfile نام فایل Dockerfile مربوط به ساخت تصویر را مشخص می‌کند.

ports: - 3100:3000: پورت 3000 داخل کانتینر را به پورت 3100 در میزبان نگاشت می‌دهد.

depends_on: - backend: با این دستور، اعلام می‌شود که سرویس فرانت‌اند باید بعد از راه‌اندازی سرویس بک‌اند (backend) شروع به کار کند و به آن وابسته است.

در متغیرREACT_APP_API_BASE_URL ، این تنظیم به فرانت‌اند اطلاع می‌دهد که آدرس API بک‌اند در محیط توسعه به "http://backend:5000"است. این نام "backend" به دلیل استفاده از شبکه‌های Docker به عنوان نام سرویس مرتبط با بک‌اند در همان فایل Docker Compose است.

تنظیمات سرویس فرانت‌اند

یک کانتینر با user- frontend-app ایجاد می‌شود و از تصویر user-frontend-app:1.0 استفاده می‌شود. متغیرهای محیطی NODE_ENV و REACT_APP_API_BASE_URL تنظیم شده‌اند و برای ساخت کانتینر از Dockerfile در مسیر user-portal-frontend استفاده می‌شود. پورت 3000 داخل کانتینر به پورت 3200 میزبان متصل شده و به سرویس‌های بک‌اند و فرانت‌اند وابسته است تا پس از آماده‌سازی، اجرا شود. همچنین یک والیوم برای ذخیره داده‌ها ایجاد شده است.

```
47  frontend:
48    container_name: frontend-app
49    image: frontend-app:1.0
50    environment:
51      - NODE_ENV=docker
52      - REACT_APP_API_BASE_URL=http://backend:5000
53
54    build:
55      context: frontend
56      dockerfile: Dockerfile
57    ports:
58      - 3100:3000
59    depends_on:
60      - backend
61
```

در این بخش از فایل Docker Compose ما، یک سرویس جدید به نام user-frontend ایجاد نمودیم که یک نسخه مخصوص از فرانت‌اند برای کاربران است. از جهت وابستگی این سرویس به توابع backend و frontend وابسته بوده همچنین، یک تنظیمات Volume برای ذخیره داده‌های MongoDB به نام mongo-data نیز در اینجا تعریف شده است .

user-frontend

container_name: user-frontend-app نام کانتینر user-frontend را به "user-frontend-app" تعیین می‌کند.

image: user-frontend-app:1.0 تصویر Docker مورد استفاده برای سرویس را مشخص می‌کند.

environment متغیرهای محیطی را برای تنظیم محیط اجرای user-frontend مشخص می‌کند. در اینجا NODE_ENV به "docker" و REACT_APP_API_BASE_URL به "http://backend:5000" تنظیم شده‌اند.

Build تنظیمات مربوط به ساخت تصویر را مشخص می‌کند .context به معنای مسیر محل فایل‌های ساخت است و dockerfile نام فایل Dockerfile مربوط به ساخت تصویر را مشخص می‌کند.

ports: - 3200:3000: پورت 3000 داخل کانتینر را به پورت 3200 در میزبان نگاشت می‌دهد .

depends_on: - backend - frontend اعلام می‌شود که سرویس user-frontendباید بعد از راه‌اندازی سرویس‌های backend و frontend شروع به کار کند.

در مورد:volumes

mongo-data یک والیوم به نام mongo-data با درایور local ایجاد می‌کند. این ولوم برای ذخیره داده‌های پایدار MongoDB استفاده می‌شود.

```
62    user-frontend:
63      container_name: user-frontend-app
64      image: user-frontend-app:1.0
65      environment:
66        - NODE_ENV=docker
67        - REACT_APP_API_BASE_URL=http://backend:5000
68      build:
69        context: user-portal-frontend
70        dockerfile: Dockerfile
71
72      ports:
73
74        - 3200:3000
75
76
77      depends_on:
78        - backend
79        - frontend
80
81  volumes:
82    mongo-data:
83      driver: local
```

پس از اجرای پروژه به صورت Local و اجرای داکر می توان از طریق مسیرهای زیر سامانه های کاربری و ادمین قابل دسترسی است:

Backend server : localhost:5000/

Admin Frontend : localhost:3100/

User Frontend : localhost:3200/

پنل ادمین

در صفحه اول باید با نام کاربری و رمز عبور مدیر احراز هویت شود. به عنوان پیشفرض یوزر نیم و پسورد به صورت ("sysadmin","systemadmin") تعریف شده است.

سیستم مدیریتی ما دارای بخش هایی به شرح زیر است، توضیحاتی برای ویژگی‌های اصلی این وب سایت آورده شده است:

ثبت نام معلمان:

معلمان قبل از استفاده از سرویس، می‌توانند از قسمت ثبت‌نام، اطلاعات شخصی خود را وارد کنند.

بعد از تایید مدیر سیستم، حساب کاربری معلم ایجاد می‌شود.

نام درس:

معلمان می‌توانند درس‌های خود را با نام‌ها و اطلاعات توضیحی ثبت کنند.

هر درس می‌تواند شناسه یکتا داشته باشد که برای ارتباط با دیگر اجزای سیستم استفاده می‌شود.

آزمون آنلاین:

ذخیره اطلاعات دانش‌آموزان در پایگاه داده با اطلاعاتی نظیر نام، نام‌خانوادگی، شماره دانش‌آموزی و اطلاعات تماس.

معلمان می‌توانند نمرات و عملکرد دانش‌آموزان را به صورت آنلاین ثبت و مشاهده کنند.

گزارش:

امکان بررسی نتایج آزمون‌ها و عملکرد دانش‌آموزان برای معلمان.

ایجاد گزارش‌های سفارشی بر اساس دسته‌بندی‌های مختلف.

امنیت:

استفاده از سیستم امنیتی برای حفاظت از اطلاعات حساب‌های کاربری و داده‌ها.

امکان اجرای اتصال امن برای رمزنگاری اطلاعات در ارتباطات با سایت.

پشتیبانی فنی:

ایجاد سیستم پشتیبانی فنی برای حل مشکلات و پاسخ به سوالات معلمان و دانش‌آموزان.

با این ویژگی‌ها، وب سایت پنل ادمین برای آزمون آنلاین قادر به ارتقاء تجربه آموزشی و مدیریت آزمون‌ها خواهد بود. معلمان می‌توانند آزمون‌های آنلاین را بسازند.امکان تنظیم سوالات مختلف، انتخاب گزینه‌ها، و تعیین امتیاز برای هر سوال.

زمان‌بندی آزمون و ایجاد قوانین خاص نیز ممکن است.

پایگاه داده دانش‌آموزان:

ذخیره اطلاعات دانش‌آموزان در پایگاه داده با اطلاعاتی نظیر نام، نام‌خانوادگی، شماره دانش‌آموزی و اطلاعات تماس.

معلمان می‌توانند نمرات و عملکرد دانش‌آموزان را به صورت آنلاین ثبت و مشاهده کنند.

گزارش:

امکان بررسی نتایج آزمون‌ها و عملکرد دانش‌آموزان برای معلمان. ایجاد گزارش‌های سفارشی بر اساس دسته‌بندی‌های مختلف.

امنیت:

استفاده از سیستم امنیتی برای حفاظت از اطلاعات حساب‌های کاربری و داده‌ها.

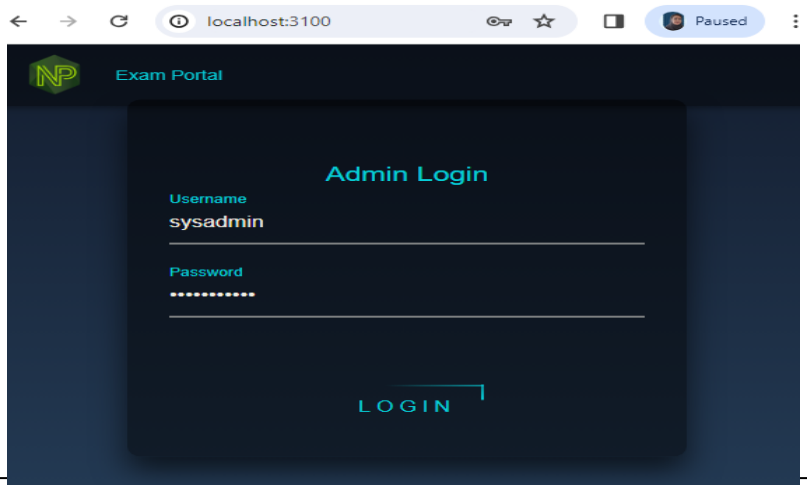
امکان اجرای اتصال امن برای رمزنگاری اطلاعات در ارتباطات با سایت.

پشتیبانی فنی:

ایجاد سیستم پشتیبانی فنی برای حل مشکلات و پاسخ به سوالات معلمان و دانش‌آموزان.

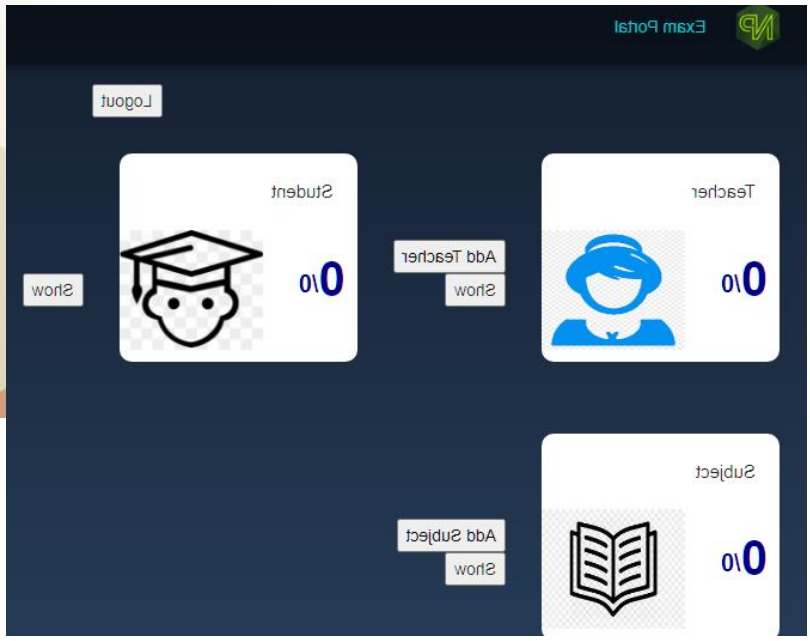
با این ویژگی‌ها، وب سایت پنل ادمین برای آزمون آنلاین قادر به ارتقاء تجربه آموزشی و مدیریت آزمون‌ها خواهد بود.

صفحه اولیه پنل مدیریتی :



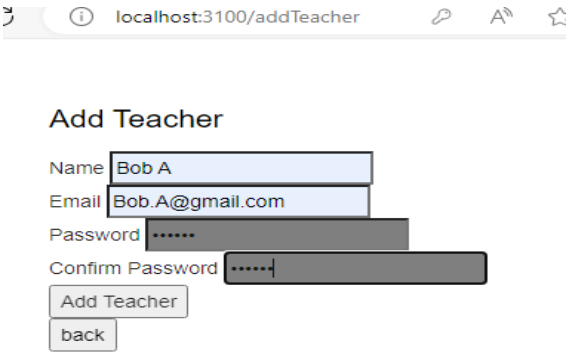
و

ورود به پنل مدیریتی و امکانات

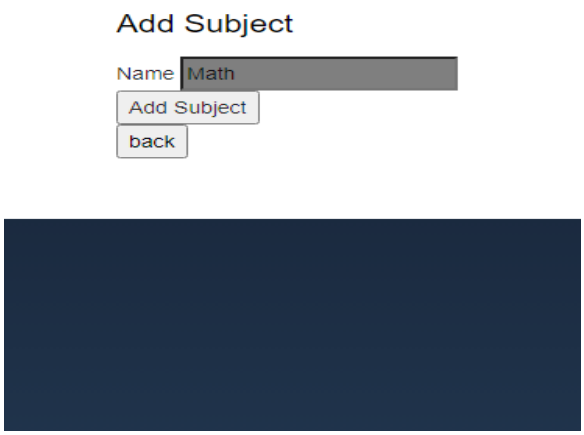


پس از ورود با حساب ادمینی همانطور که قابل مشاهده است امکان اضافه کردن معلم، دسترسی به پایگاه داده دانش آموزی و افزودن درس از اجزای اصلی سیستم بوده و قابل اجرا می باشد.

ایجاد حساب کاربری معلم

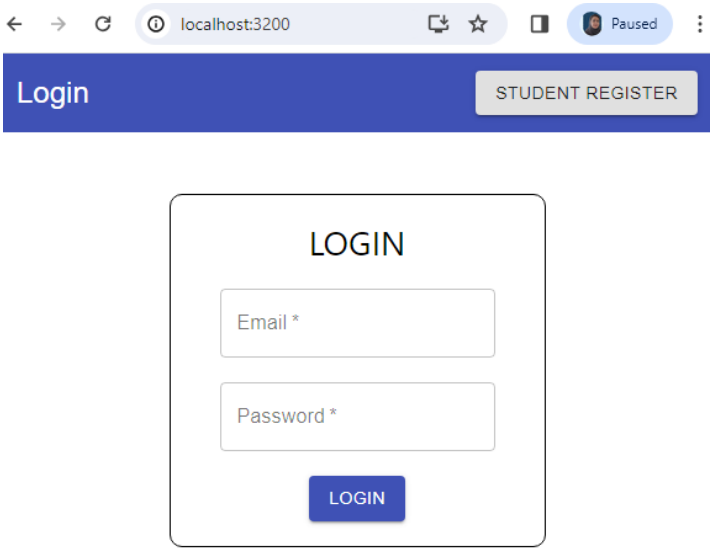


تعریف درس جدید

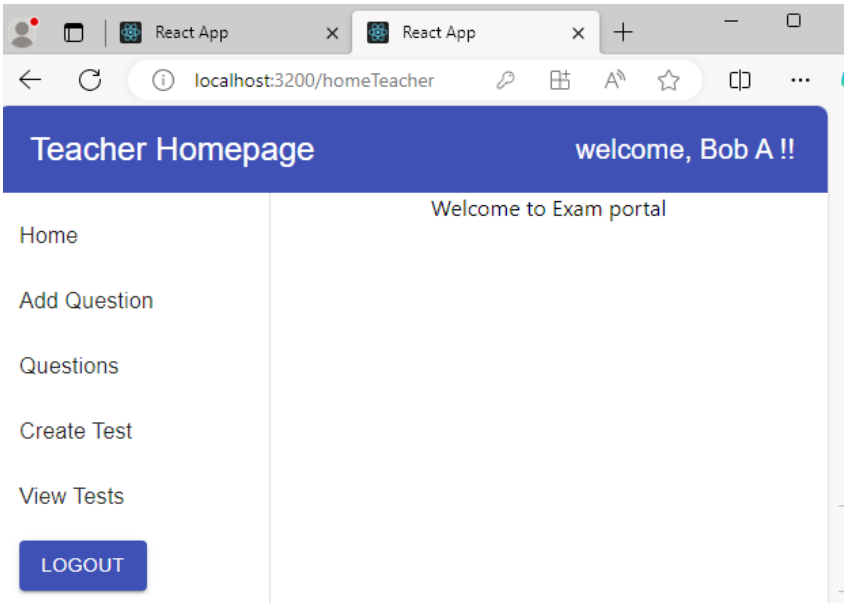


پنل کاربر

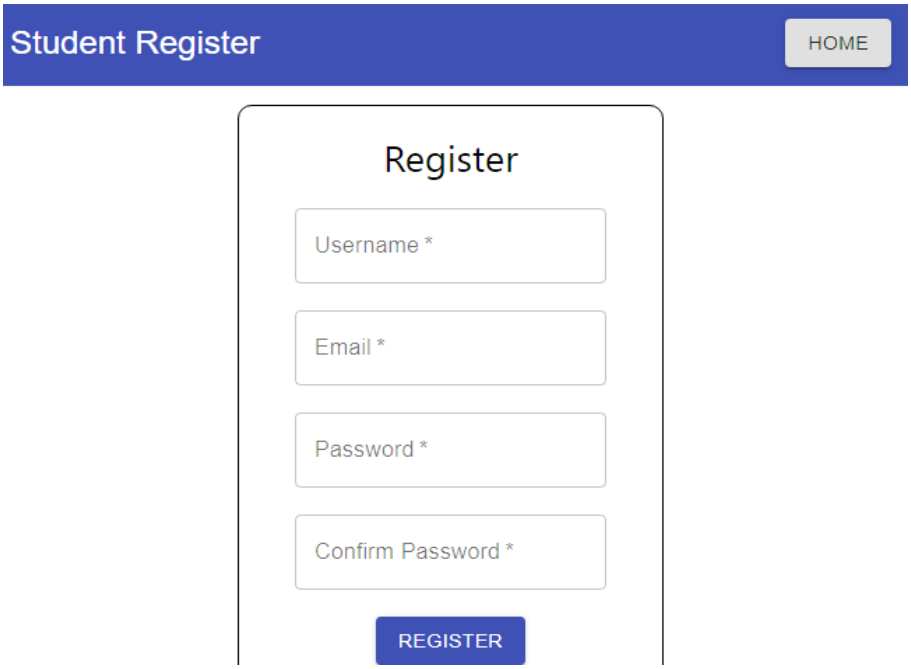
در پنل کاربر امکان ورود معلمان و دانش آموزان طبق بانک اطلاعاتی وجود دارد. همچنین ثبت نام دانش آموز جدید از این پنل انجام می شود.



ورود معلم



ثبت نام دانش آموز



نصب سرور Mongo

git clone https://github.com/elahehsadatkhodaparast/EMS

مسیر ذخیره local

cd project-directory

نصب npm

cd backend
npm install
cd ../frontend
npm install
cd ../user-portal-frontend
npm install
Start the backend server

cd backend
npm start
Start the frontend client for admin

cd frontend
npm start
Start the frontend client for teacher/student

cd user-portal-frontend
npm start

اجرا داکر

Run With Docker
build docker images
docker-compose build
#Run container and services
docker-compose up