Final Lab SCD

Course Instructor
Junaid Ali

Lab Instructor
Noor ul Ain

Submitted By:
22i-8761 Muhammad Bin Tariq

Date: 24-05-2025

Spring 2025

Department of Computer Science

FAST – National University of Computer & Emerging Science

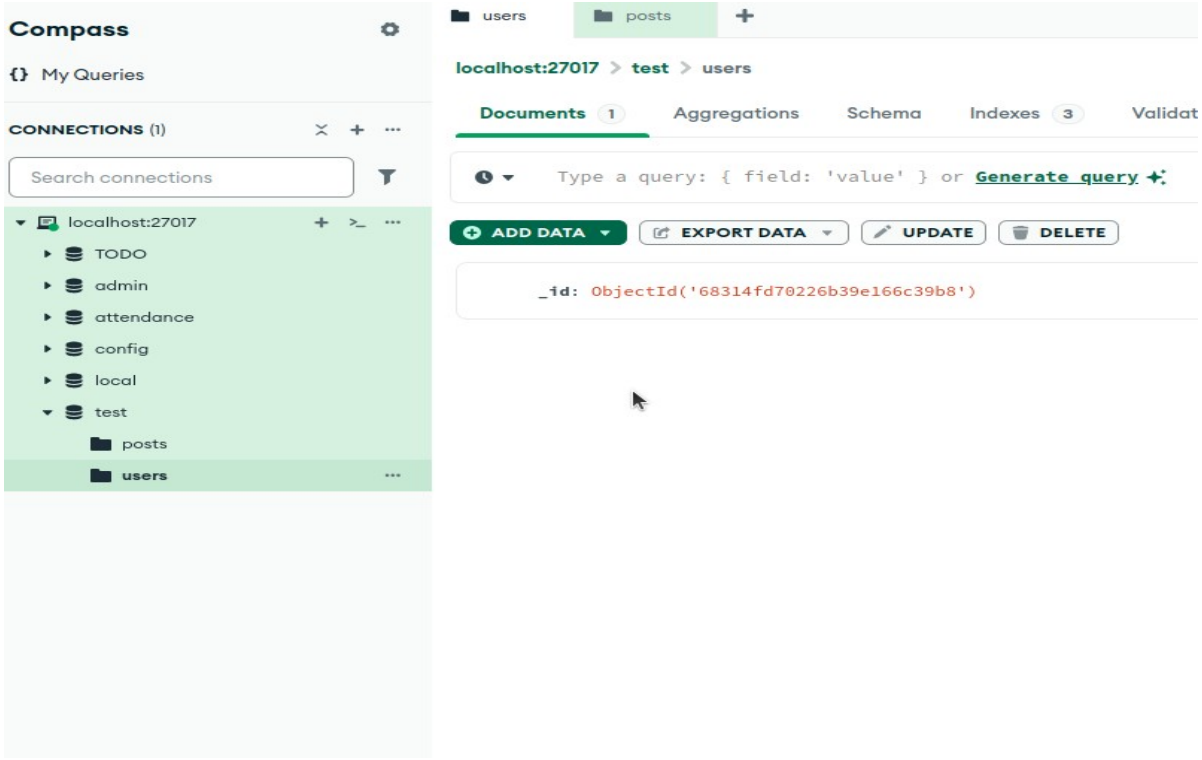
Question 1:

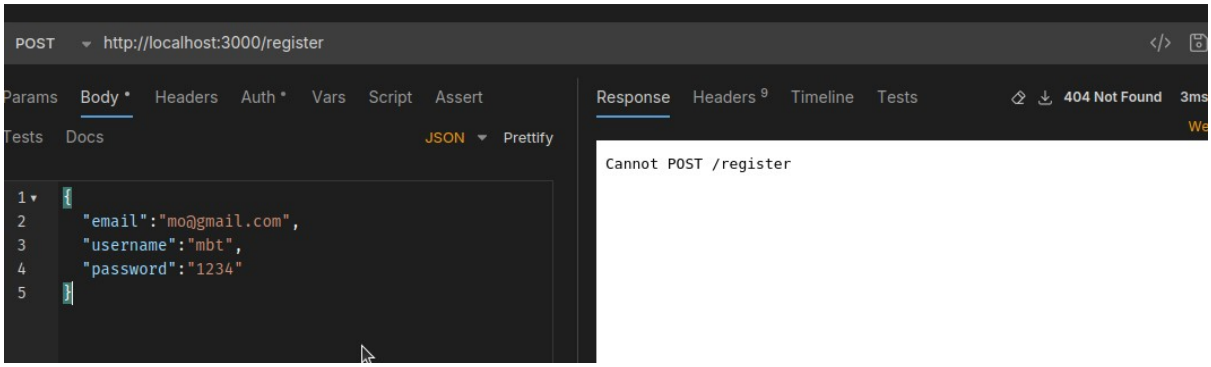
- Start the service by defining the Mongo Url and port in .env file.

```
.env
1   MONGO_URI=mongodb://localhost:27017/test
2   PORT=3000
3
```

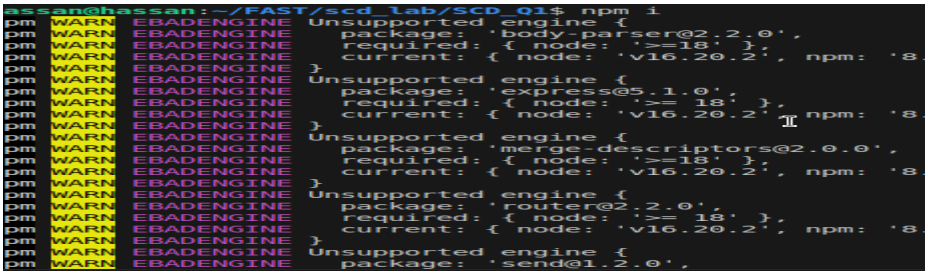
Auth Service:

```
// Register User
exports.register = async (req, res) => {
  const { username, email, password } = req.body;

  try {
    const user = new User({
      username,
      email,
      password: await bcrypt.hash(password, 10),
    });
    await user.save();
    res.status(201).json({ message: "User registered successfully" });
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
};

// Login User
exports.login = async (req, res) => {
  const { email, password } = req.body;
  try {
    const user = await User.findOne({ email });
    if (!user)
      return res.status(400).json({ message: "Invalid email or password" });

    const match = await bcrypt.compare(password, user.password);
    if (!match)
      return res.status(400).json({ message: "Invalid email or password" });
  } catch {
    return res.status(400).json({ message: "Invalid credentials" });
  }
```



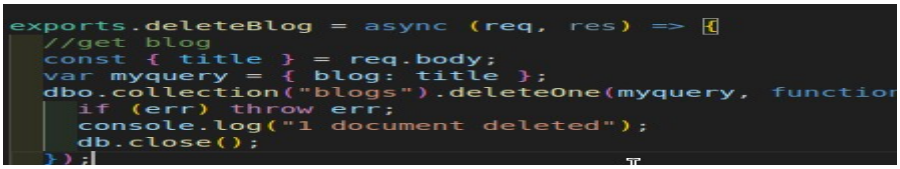


NOTE: This is not posting. Tried everything maybe its due to my node version i.e 16



Blog Service:

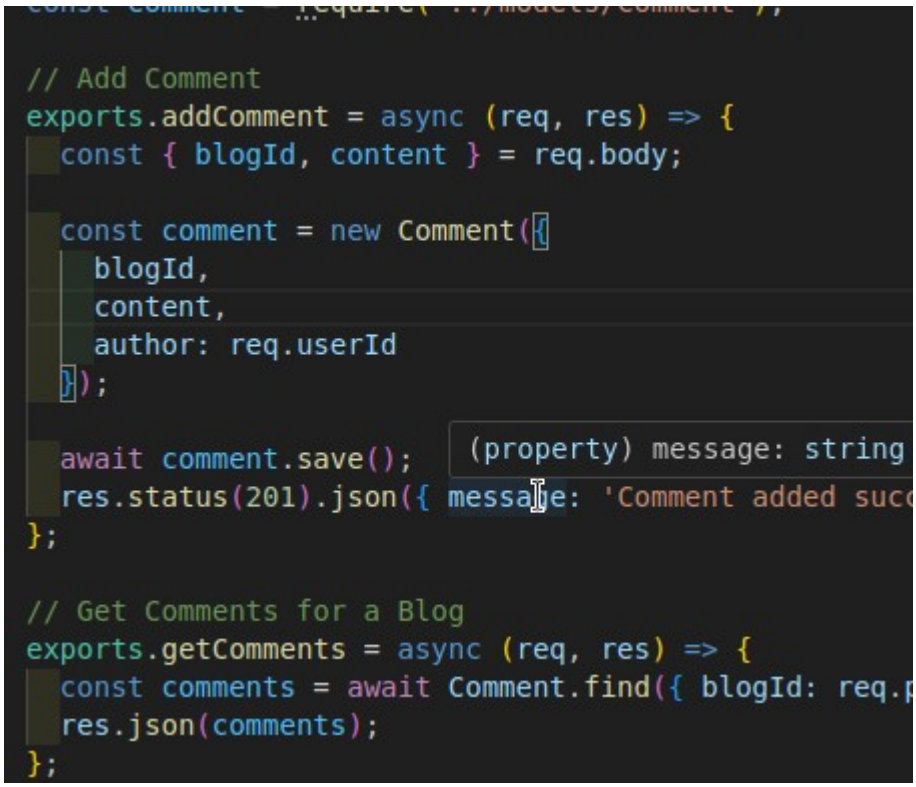
created a deleted function, create and view were already present.



Comment Service:

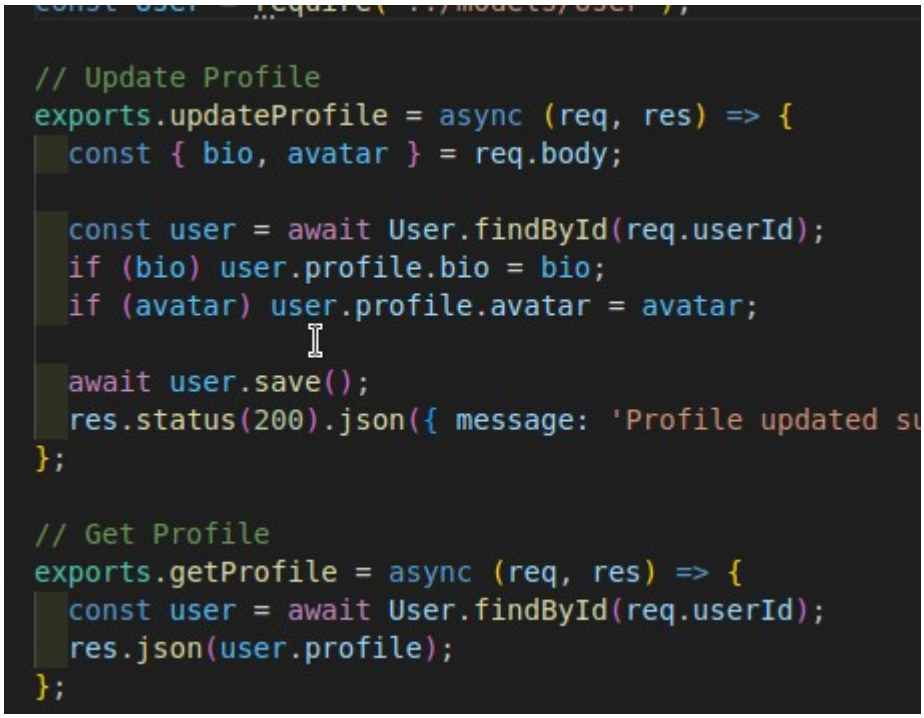
Functionality implemented

JWT is required to login so incorrect login would automatically not upload comment



Profile Services:

Implemented already



Api Gateway:

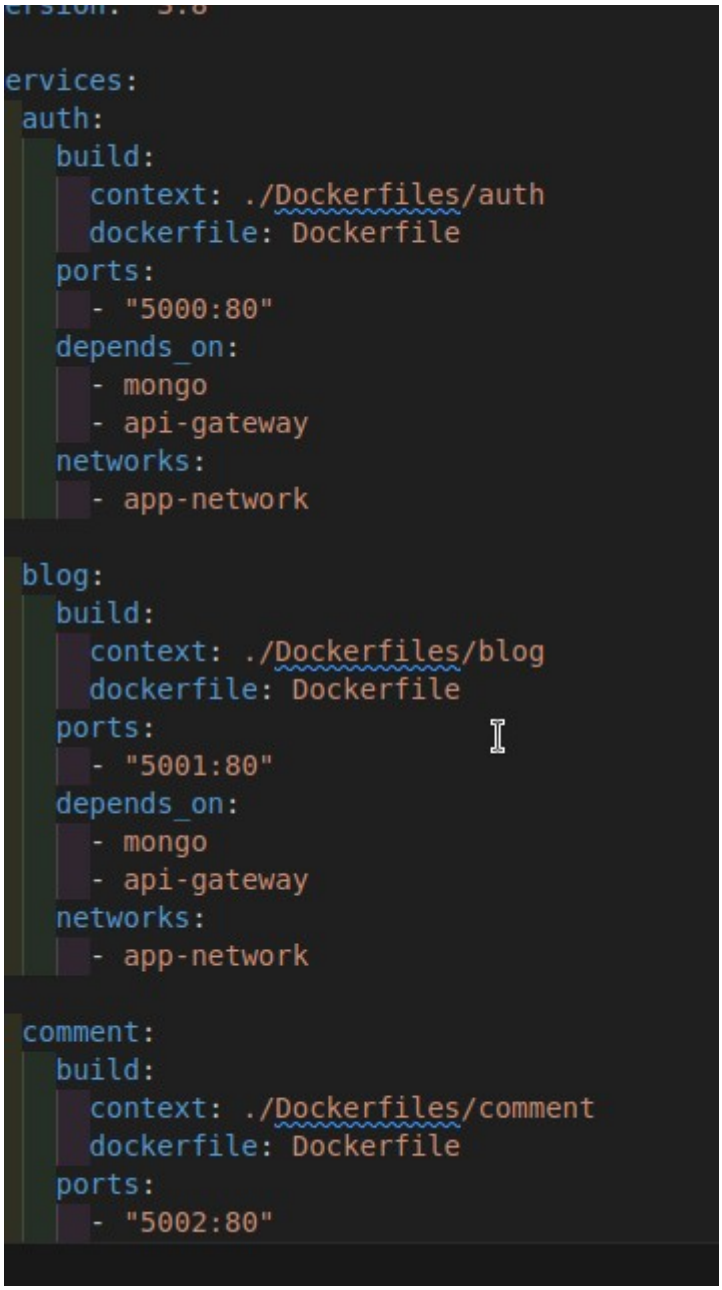
```
// API gateway
const express = require("express");
const httpProxy = require("express-http-proxy");
const app = express();

const AuthService = httpProxy("http://localhost:5001&quot;");

const BlogService = httpProxy("http://localhost:5002&quot;");

const CommentService = httpProxy("http://localhost:5003&quot;");

const ProfileService = httpProxy("http://localhost:5004&quot;");

app.use(express.json());
                (parameter) req: Request<{}, any, any, qs.ParsedQs, Record<strin

app.use("/auth", (req, res, next) => {
  AuthService(req, res, next);
});

app.use("/blog", (req, res, next) => {
  BlogService(req, res, next);
});

app.use("/comments", (req, res, next) => {
  bookingServiceProxy(req, res, next);
});

app.use("/profile", (req, res, next) => {
  bookingServiceProxy(req, res, next);
});
```

Part B

Dockerfiles of each services (only pasted one)

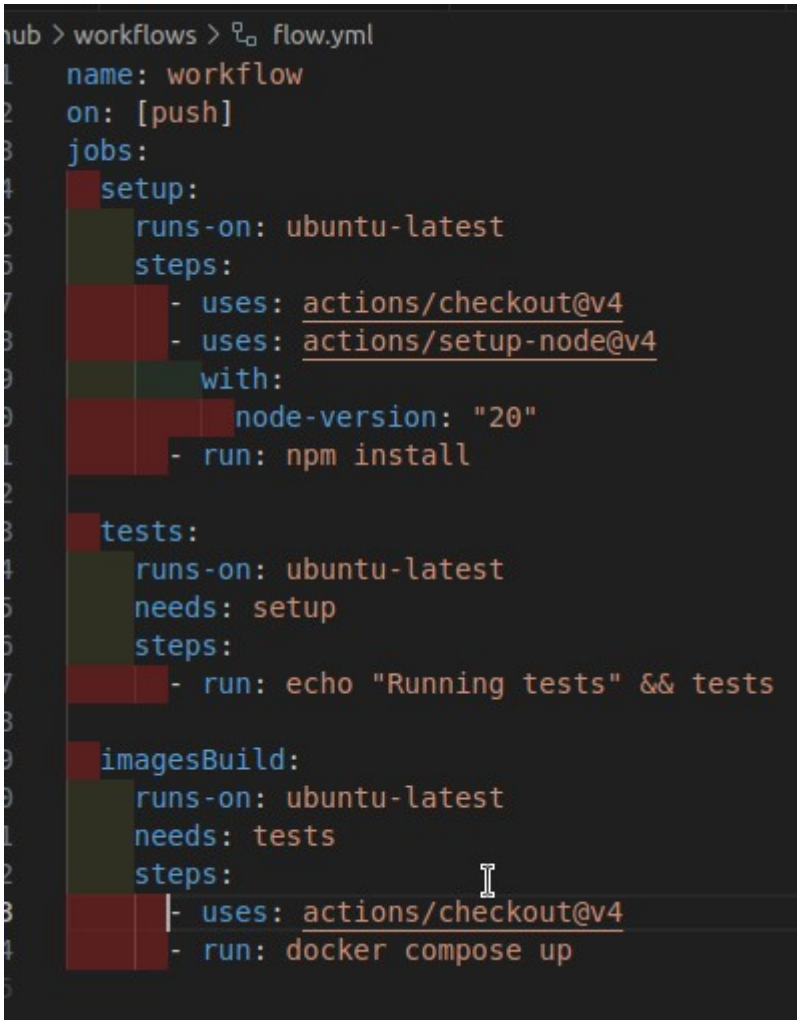
```
# Use Node.js base image
FROM node:18-alpine

# Set working directory
WORKDIR /auth

# Copy package.json and package-lock.json
COPY ../../package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the app files
COPY . .

# Expose the backend port
EXPOSE 5000
```

Docker compose



```
ervices:
 auth:
   build:
     context: ./Dockerfiles/auth
     dockerfile: Dockerfile
   ports:
     - "5000:80"
   depends_on:
     - mongo
     - api-gateway
   networks:
     - app-network

 blog:
   build:
     context: ./Dockerfiles/blog
     dockerfile: Dockerfile
   ports:
     - "5001:80"
   depends_on:
     - mongo
     - api-gateway
   networks:
     - app-network

 comment:
   build:
     context: ./Dockerfiles/comment
     dockerfile: Dockerfile
   ports:
     - "5002:80"
```

Part C:


hub > workflows > flow.yml
name: workflow
on: [push]
jobs:
setup:
runs-on: ubuntu-latest
steps:
- uses: actions/checkout@v4
- uses: actions/setup-node@v4
with:
node-version: "20"
- run: npm install
tests:
runs-on: ubuntu-latest
needs: setup
steps:
- run: echo "Running tests" && tests
imagesBuild:
runs-on: ubuntu-latest
needs: tests
steps:
- uses: actions/checkout@v4
- run: docker compose up

Question 2:

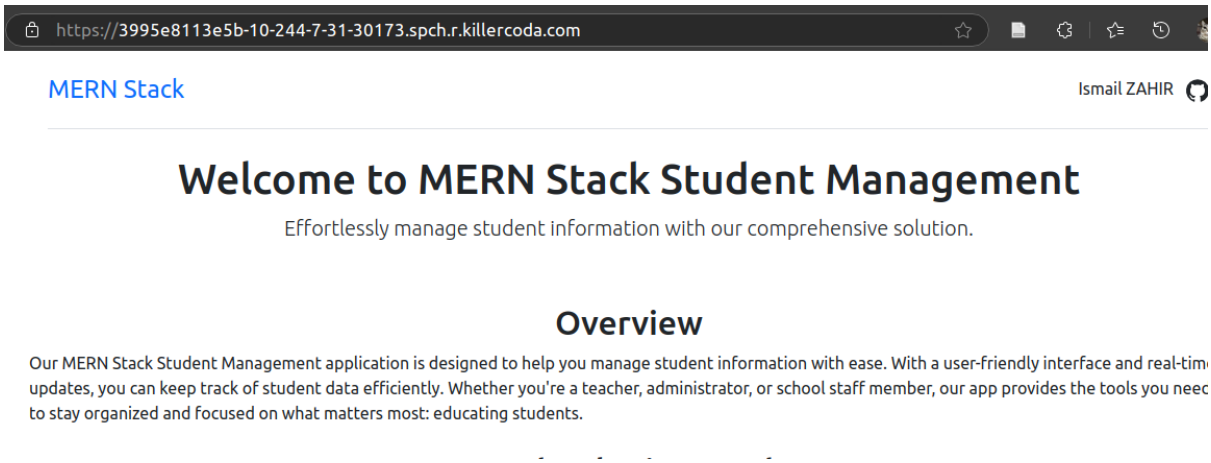
Task 1

```
controlplane:~$ kubectl create namespace dataviz-ns
namespace/dataviz-ns created
```

Task 2

Frontend:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: frontend-deployment
  template:
    metadata:
      labels:
        app: frontend-deployment
    spec:
      containers:
        - name: react
          image: ismailza/mern-stack-app-frontend:latest
          ports:
            - containerPort: 5173
```



Backend:

frontend.yml U | backend.yaml U | mongo.yaml U | pv-pvc.yaml U

backend.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: backend-deployment
  template:
    metadata:
      labels:
        app: backend-deployment
    spec:
      containers:
        - name: react
          image: ismailza/mern-stack-app-backend:latest
          ports:
            - containerPort: 3000
---
apiVersion: v1
```

```
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
backend-service    NodePort    10.104.80.204   <none>        3000:30300/TCP   5s
frontend-service   NodePort    10.96.53.51     <none>        5173:30173/TCP   7m59s
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP          12d
```

Mongo:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongo
  template:
    metadata:
      labels:
        app: mongo
```

PVC

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mongodb-pv
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /data/mongodb
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongodb-pvc
spec:
  storageClassName: ""
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

```
controlplane:~$ kubectl apply -f pv-pvc.yaml
persistentvolume/mongodb-pv created
persistentvolumeclaim/mongodb-pvc created
controlplane:~$ nano mongo.yaml
controlplane:~$ kubectl apply -f mongo.yaml
deployment.apps/mongo created
```

Task 3:

```
      - containerPort: 3000
initContainers:
  - name: busybox-init
    image: busybox
    command: ["sh", "c", 'echo "Initializing Backend ... " && sleep 10']
```

Task 4:

```
- name: sidecar
  image: busybox
  command: ["sh", "c", 'echo "Logs : " >> /var/log/app.log ']

  volumeMount:
    - name: shared-logs
  mountPath: /var/log

  volumes:
    -name: shared-logs
    emptyDir: {}
```

Task 5

```
controlplane:~$ kubectl create configmap backend-config --from-literal=APP_MODE=production
configmap/backend-config created
controlplane:~$ kubectl create secret backend-secret --from-literal=DB_User=admin
error: unknown flag: --from-literal
See 'kubectl create secret --help' for usage.
controlplane:~$ kubectl create secret generic backend-secret --from-literal=DB_User=admin
secret/backend-secret created
controlplane:~$ kubectl get configmap backend-config -o yaml
apiVersion: v1
data:
  APP_MODE: production
kind: ConfigMap
metadata:
  creationTimestamp: "2025-05-24T05:45:31Z"
  name: backend-config
  namespace: default
  resourceVersion: "14923"
  uid: 7a738fbe-ac7f-4911-89a0-3a87fd1af204
controlplane:~$ kubectl get secret backend-secret -o yaml
apiVersion: v1
data:
  DB_User: YWRtaW4=
kind: Secret
metadata:
  creationTimestamp: "2025-05-24T05:46:44Z"
  name: backend-secret
  namespace: default
  resourceVersion: "15033"
  uid: 5dec84e2-5866-432a-b2c9-103001c2874d
type: Opaque
```

Task 6

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mongodb-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /data/mongodb
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongodb-pvc
spec:
  storageClassName: ""
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

PVC for the frontend

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: frontend-pvc
labels:
name: frontend-pvc
spec:
storageClassName: "standard"

accessModes:
- ReadWriteOnce
resources:
requests:
storage: 200Mi
```

```yaml
    app: frontend-deployment
spec:
  containers:
    - name: react
      image: ismailza/mern-stack-app-frontend:latest
      ports:
        - containerPort: 5173
      initContainers: ##inint for logs

    - name: busybox-init
      image: busybox
      command: ["sh", "c", 'echo "Logs " >> /var/log/app.log']
      volumeMount:
        persistentVolumeClain: frontend-pvc
        mountPath: /use/share/nginx/html
```

Task 7

```yaml
    resources:
      requests:
        cpu: "100m"
        memory: "64Mi"
      limits:
        cpu: "200m"
        memory: "128Mi"
```

Task 8

NOTE: **Created a new session here**

```
spec:
  containers:
  - name: react
  image: ismailza/mern-stack-app-backend:1.0.1
  ports: - containerPort: 3000
```

```
strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
```

Applying

```
controlplane:~$ kubectl apply -f backend.yml
deployment.apps/backend-deployment created
service/backend-service unchanged
controlplane:~$ kubectl set image deployement
```