

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.

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

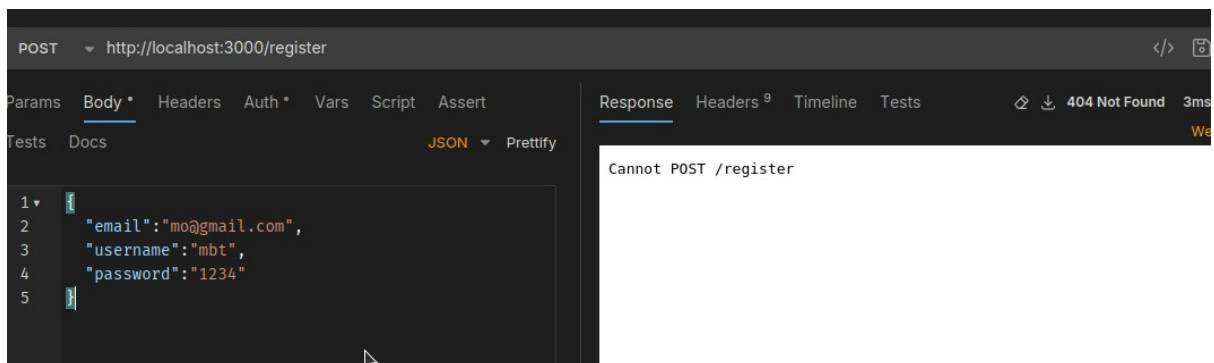
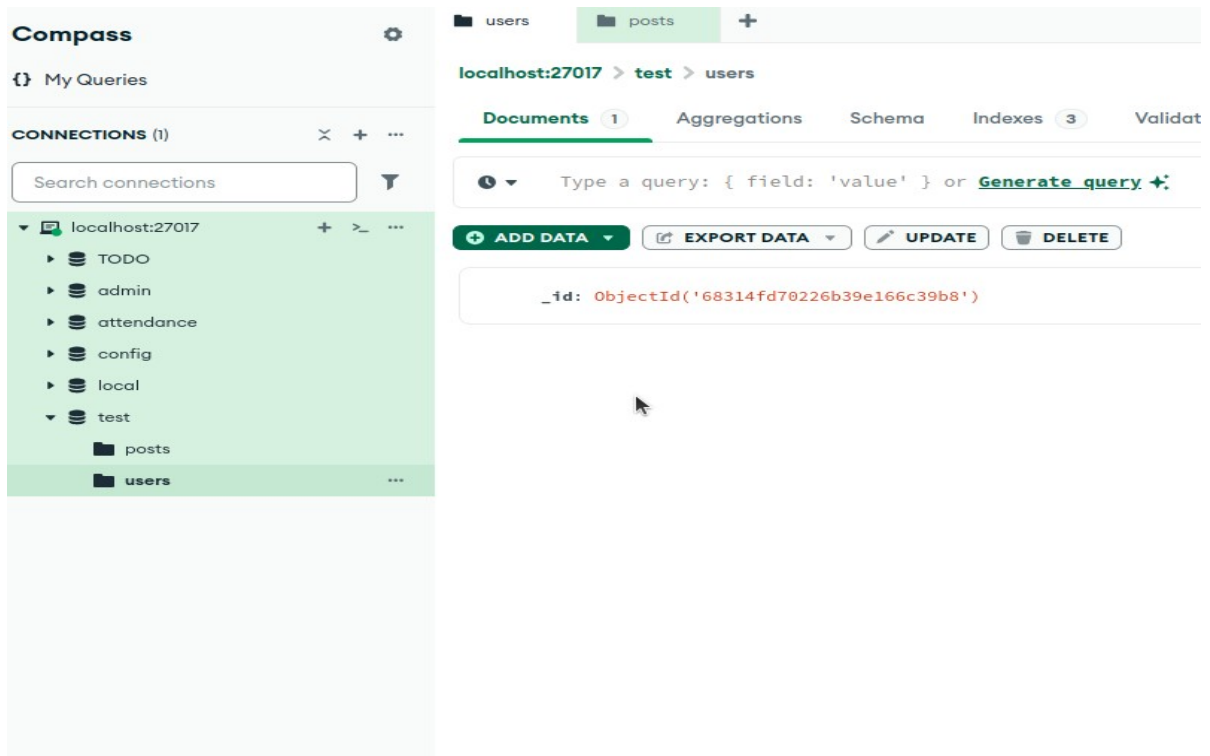
Auth Service:

```
// 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

```

assan@hassan:~/FAST/scd_lab/SCD_01$ npm i
pm WARN EBADENGINE Unsupported engine {
pm WARN EBADENGINE   package: 'body-parser@2.2.0',
pm WARN EBADENGINE   required: { node: '>=18' },
pm WARN EBADENGINE   current: { node: 'v16.20.2', npm: '8.19.4' }
pm WARN EBADENGINE }
pm WARN EBADENGINE Unsupported engine {
pm WARN EBADENGINE   package: 'express@5.1.0',
pm WARN EBADENGINE   required: { node: '>=18' },
pm WARN EBADENGINE   current: { node: 'v16.20.2', npm: '8.19.4' }
pm WARN EBADENGINE }
pm WARN EBADENGINE Unsupported engine {
pm WARN EBADENGINE   package: 'merge-descriptors@2.0.0',
pm WARN EBADENGINE   required: { node: '>=18' },
pm WARN EBADENGINE   current: { node: 'v16.20.2', npm: '8.19.4' }
pm WARN EBADENGINE }
pm WARN EBADENGINE Unsupported engine {
pm WARN EBADENGINE   package: 'router@2.2.0',
pm WARN EBADENGINE   required: { node: '>=18' },
pm WARN EBADENGINE   current: { node: 'v16.20.2', npm: '8.19.4' }
pm WARN EBADENGINE }
pm WARN EBADENGINE Unsupported engine {
pm WARN EBADENGINE   package: 'send@1.2.0',

```

Blog Service:

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

```

exports.deleteBlog = async (req, res) => {
  //get blog
  const { title } = req.body;
  var myquery = { blog: title };
  dbo.collection("blogs").deleteOne(myquery, function (err, deletedCount) {
    if (err) throw err;
    console.log("1 document deleted");
    db.close();
  });
};

```

Comment Service:

Functionality implemented

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

```
const Comment = require('../models/comment');

// Add Comment
exports.addComment = async (req, res) => {
  const { blogId, content } = req.body;

  const comment = new Comment({
    blogId,
    content,
    author: req.userId
  });

  await comment.save();
  res.status(201).json({ message: 'Comment added successfully' });

  // Get Comments for a Blog
  exports.getComments = async (req, res) => {
    const comments = await Comment.find({ blogId: req.params.blogId });
    res.json(comments);
  };
};
```

Profile Services:

Implemented already

```
const User = require('../models/user');

// Update Profile
exports.updateProfile = async (req, res) => {
  const { bio, avatar } = req.body;

  const user = await User.findById(req.userId);
  if (bio) user.profile.bio = bio;
  if (avatar) user.profile.avatar = avatar;

  await user.save();
  res.status(200).json({ message: 'Profile updated successfully' });

  // Get Profile
  exports.getProfile = async (req, res) => {
    const user = await User.findById(req.userId);
    res.json(user.profile);
  };
};
```

Api Gateway:

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

const AuthService = httpProxy("http://localhost:5001");
const BlogService = httpProxy("http://localhost:5002");
const CommentService = httpProxy("http://localhost:5003");
const ProfileService = httpProxy("http://localhost:5004");

app.use(express.json());
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

```
version: 3.8

services:
  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
1  name: workflow
2  on: [push]
3  jobs:
4    setup:
5      runs-on: ubuntu-latest
6      steps:
7        - uses: actions/checkout@v4
8        - uses: actions/setup-node@v4
9          with:
10           node-version: "20"
11        - run: npm install
12
13    tests:
14      runs-on: ubuntu-latest
15      needs: setup
16      steps:
17        - run: echo "Running tests" && tests
18
19    imagesBuild:
20      runs-on: ubuntu-latest
21      needs: tests
22      steps:
23        - uses: actions/checkout@v4
24        - run: docker compose up
```

Question 2:

Task 1

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

Task 2

Frontend:

```
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
```

<https://3995e8113e5b-10-244-7-31-30173.spch.r.killercode.com>

MERN Stack

Ismail ZAHIR

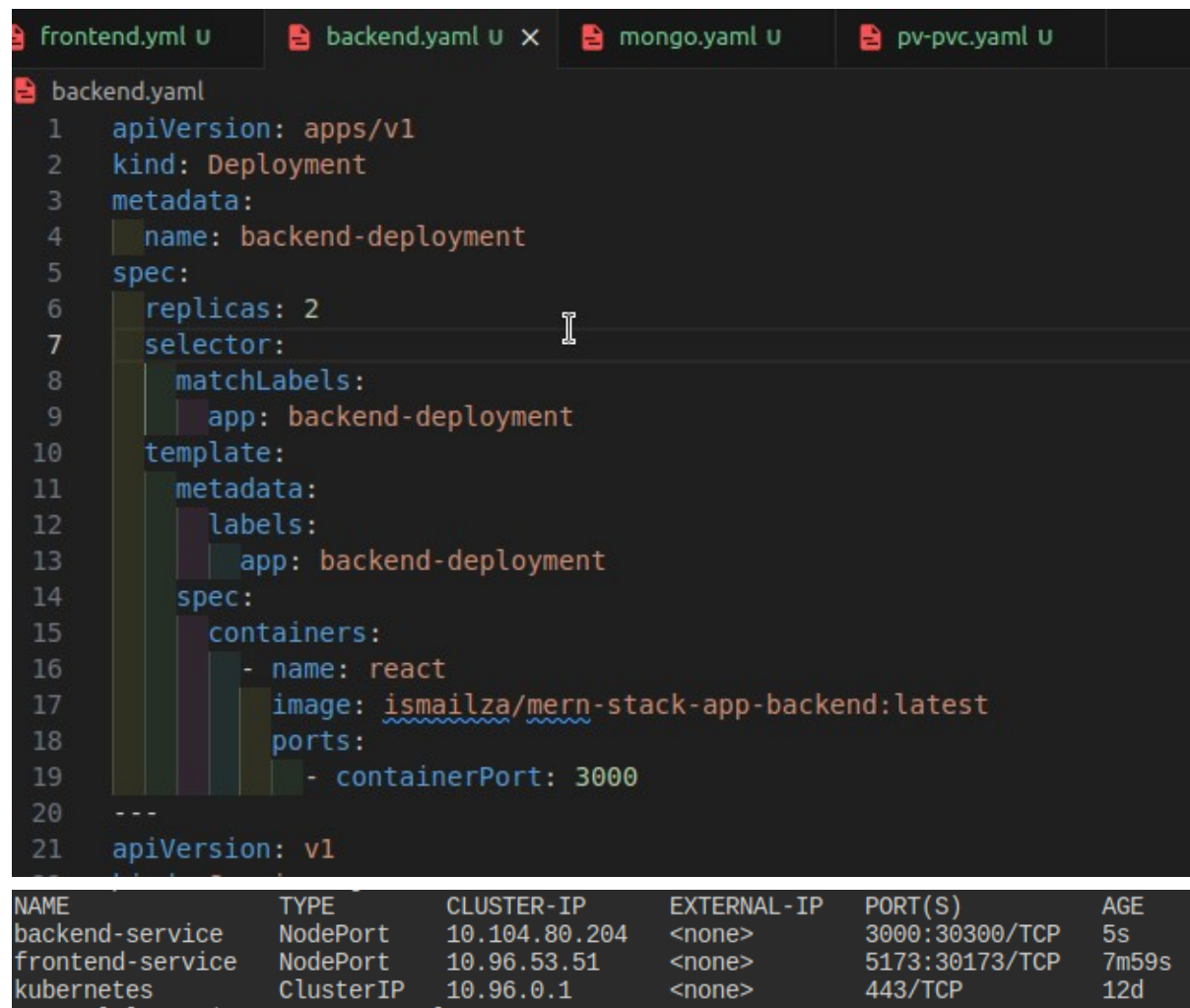
Welcome to MERN Stack Student Management

Effortlessly manage student information with our comprehensive solution.

Overview

Our MERN Stack Student Management application is designed to help you manage student information with ease. With a user-friendly interface and real-time updates, you can keep track of student data efficiently. Whether you're a teacher, administrator, or school staff member, our app provides the tools you need to stay organized and focused on what matters most: educating students.

Backend:



The screenshot shows a code editor with four tabs: 'frontend.yml u', 'backend.yaml u x', 'mongo.yaml u', and 'pv-pvc.yaml u'. The 'backend.yaml' tab is active, displaying a YAML configuration for a Deployment. The configuration includes metadata (name: backend-deployment), spec (replicas: 2, selector: matchLabels: app: backend-deployment), and a template (metadata: labels: app: backend-deployment, spec: containers: - name: react, image: ismailza/mern-stack-app-backend:latest, ports: - containerPort: 3000). Below the code editor, a table lists Kubernetes resources.

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:

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

PVC

```

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
controlplane:~$

```

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

```
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

```
1  apiVersion: v1
2  kind: PersistentVolumeClaim
3  metadata:
4  name: frontend-pvc
5  labels:
6  name: frontend-pvc
7  spec:
8  storageClassName: "standard"
9
10 accessModes:
11 - ReadWriteOnce
12 resources:
13 requests:
14 storage: 200Mi
```

```
  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:
        persistentVolumeClaim: frontend-pvc
        mountPath: /use/share/nginx/html
```

Task 7

```
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 deployment
```