Experiment No.10

Date:0.05.25

Aim: Demonstrate implementation of Reactive Services.

CO Mapping – CO 4

Objective:

1. To develop program based on Reactive Services

Code:

```
/* ======================= MODELS ======================== */

// src/main/java/com/spring_reactive/reactive/model/Job.java
package com.spring_reactive.reactive.model;

import org.springframework.data.annotation.Id;
import org.springframework.data.annotation.Transient;
import org.springframework.data.mongodb.core.mapping.Document;
import com.spring_reactive.reactive.util.DateUtil;

import java.time.LocalDateTime;
import java.util.Date;
import java.util.List;

@Document(collection = "jobs")
public class Job {
    @Id
    private String id;
```

```
    private String title;
    private String company;
    private String description;
    private List<String> requirements;
    private String location;
    private String jobType; // FULL_TIME, PART_TIME, CONTRACT
    private Double salary;
    private LocalDateTime postedDate;
    private LocalDateTime expiryDate;
    private boolean active;

    public Job() {
    }

    public Job(String title, String company, String description, List<String> requirements,
            String location, String jobType, Double salary, LocalDateTime postedDate,
            LocalDateTime expiryDate, boolean active) {
        this.title = title;
        this.company = company;
        this.description = description;
        this.requirements = requirements;
        this.location = location;
        this.jobType = jobType;
        this.salary = salary;
        this.postedDate = postedDate;
        this.expiryDate = expiryDate;
        this.active = active;
```

```
    }

    // Utility methods for JSP date formatting
    @Transient
    public Date getPostedDateAsDate() {
        return DateUtil.toDate(postedDate);
    }

    @Transient
    public Date getExpiryDateAsDate() {
        return DateUtil.toDate(expiryDate);
    }

    // Getters and Setters
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
```

```
        this.title = title;
    }

    public String getCompany() {
        return company;
    }

    public void setCompany(String company) {
        this.company = company;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public List<String> getRequirements() {
        return requirements;
    }

    public void setRequirements(List<String> requirements) {
        this.requirements = requirements;
    }
```

```
    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }

    public String getJobType() {
        return jobType;
    }

    public void setJobType(String jobType) {
        this.jobType = jobType;
    }

    public Double getSalary() {
        return salary;
    }

    public void setSalary(Double salary) {
        this.salary = salary;
    }

    public LocalDateTime getPostedDate() {
        return postedDate;
    }
```

```
    public void setPostedDate(LocalDateTime postedDate) {
        this.postedDate = postedDate;
    }

    public LocalDateTime getExpiryDate() {
        return expiryDate;
    }

    public void setExpiryDate(LocalDateTime expiryDate) {
        this.expiryDate = expiryDate;
    }

    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }
}

// src/main/java/com/spring_reactive/reactive/model/JobSeeker.java
package com.spring_reactive.reactive.model;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;
```

```
import java.util.List;

@Document(collection = "job_seekers")
public class JobSeeker {
    @Id
    private String id;
    private String name;
    private String email;
    private String phoneNumber;
    private String resumeUrl;
    private List<String> skills;
    private String experienceLevel; // ENTRY, MID, SENIOR, etc.
    private String education;
    private List<String> preferredLocations;
    private List<String> appliedJobs;

    public JobSeeker() {
    }

    public JobSeeker(String name, String email, String phoneNumber, String resumeUrl,
                     List<String> skills, String experienceLevel, String education,
                     List<String> preferredLocations) {
        this.name = name;
        this.email = email;
        this.phoneNumber = phoneNumber;
        this.resumeUrl = resumeUrl;
```

```
        this.skills = skills;
        this.experienceLevel = experienceLevel;
        this.education = education;
        this.preferredLocations = preferredLocations;
    }

    // Getters and Setters
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }
```

```
    public void setEmail(String email) {
        this.email = email;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public String getResumeUrl() {
        return resumeUrl;
    }

    public void setResumeUrl(String resumeUrl) {
        this.resumeUrl = resumeUrl;
    }

    public List<String> getSkills() {
        return skills;
    }

    public void setSkills(List<String> skills) {
        this.skills = skills;
    }
```

```
    public String getExperienceLevel() {
        return experienceLevel;
    }

    public void setExperienceLevel(String experienceLevel) {
        this.experienceLevel = experienceLevel;
    }

    public String getEducation() {
        return education;
    }

    public void setEducation(String education) {
        this.education = education;
    }

    public List<String> getPreferredLocations() {
        return preferredLocations;
    }

    public void setPreferredLocations(List<String> preferredLocations) {
        this.preferredLocations = preferredLocations;
    }

    public List<String> getAppliedJobs() {
        return appliedJobs;
```

```
    }

    public void setAppliedJobs(List<String> appliedJobs) {
        this.appliedJobs = appliedJobs;
    }
}

// src/main/java/com/spring_reactive/reactive/model/JobApplication.java
package com.spring_reactive.reactive.model;

import org.springframework.data.annotation.Id;
import org.springframework.data.annotation.Transient;
import org.springframework.data.mongodb.core.mapping.Document;
import com.spring_reactive.reactive.util.DateUtil;

import java.time.LocalDateTime;
import java.util.Date;

@Document(collection = "job_applications")
public class JobApplication {
    @Id
    private String id;
    private String jobId;
    private String jobSeekerId;
    private LocalDateTime applicationDate;
    private String status; // APPLIED, REVIEWING, INTERVIEW, REJECTED, ACCEPTED
    private String coverLetter;
```

```
    // Transient fields for UI display only
    @Transient
    private Job attachedJob;

    @Transient
    private JobSeeker attachedJobSeeker;

    public JobApplication() {
    }

    public JobApplication(String jobId, String jobSeekerId, LocalDateTime applicationDate,
                          String status, String coverLetter) {
        this.jobId = jobId;
        this.jobSeekerId = jobSeekerId;
        this.applicationDate = applicationDate;
        this.status = status;
        this.coverLetter = coverLetter;
    }

    // Utility method for JSP date formatting
    @Transient
    public Date getApplicationDateAsDate() {
        return DateUtil.toDate(applicationDate);
    }

    // Getters and Setters
```

```
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getJobId() {
        return jobId;
    }

    public void setJobId(String jobId) {
        this.jobId = jobId;
    }

    public String getJobSeekerId() {
        return jobSeekerId;
    }

    public void setJobSeekerId(String jobSeekerId) {
        this.jobSeekerId = jobSeekerId;
    }

    public LocalDateTime getApplicationDate() {
        return applicationDate;
    }
```

```
    public void setApplicationDate(LocalDateTime applicationDate) {
        this.applicationDate = applicationDate;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public String getCoverLetter() {
        return coverLetter;
    }

    public void setCoverLetter(String coverLetter) {
        this.coverLetter = coverLetter;
    }

    public Job getAttachedJob() {
        return attachedJob;
    }

    public void setAttachedJob(Job attachedJob) {
        this.attachedJob = attachedJob;
```

```
    }

    public JobSeeker getAttachedJobSeeker() {
        return attachedJobSeeker;
    }

    public void setAttachedJobSeeker(JobSeeker attachedJobSeeker) {
        this.attachedJobSeeker = attachedJobSeeker;
    }
}

// src/main/java/com/spring_reactive/reactive/util/DateUtil.java
package com.spring_reactive.reactive.util;

import java.time.LocalDateTime;
import java.time.ZoneId;
import java.util.Date;

public class DateUtil {
    public static Date toDate(LocalDateTime localDateTime) {
        if (localDateTime == null) {
            return null;
        }
        return Date.from(localDateTime.atZone(ZoneId.systemDefault()).toInstant());
    }
}
```

```
/* ======================= REPOSITORIES ======================== */

// src/main/java/com/spring_reactive/reactive/repository/JobRepository.java
package com.spring_reactive.reactive.repository;

import com.spring_reactive.reactive.model.Job;
import org.springframework.data.mongodb.repository.ReactiveMongoRepository;
import org.springframework.stereotype.Repository;
import reactor.core.publisher.Flux;

@Repository
public interface JobRepository extends ReactiveMongoRepository<Job, String> {
    Flux<Job> findByActive(boolean active);
    Flux<Job> findByTitleContainingIgnoreCaseOrDescriptionContainingIgnoreCase(String
title, String description);
    Flux<Job> findByLocationContainingIgnoreCase(String location);
    Flux<Job> findByJobType(String jobType);
}

// src/main/java/com/spring_reactive/reactive/repository/JobSeekerRepository.java
package com.spring_reactive.reactive.repository;

import com.spring_reactive.reactive.model.JobSeeker;
import org.springframework.data.mongodb.repository.ReactiveMongoRepository;
import org.springframework.stereotype.Repository;
import reactor.core.publisher.Mono;
import reactor.core.publisher.Flux;
```

```
@Repository
public interface JobSeekerRepository extends ReactiveMongoRepository<JobSeeker, String>
{
    Mono<JobSeeker> findByEmail(String email);
    Flux<JobSeeker> findBySkillsContaining(String skill);
    Flux<JobSeeker> findByExperienceLevel(String level);
    Flux<JobSeeker> findByPreferredLocationsContaining(String location);
}

// src/main/java/com/spring_reactive/reactive/repository/JobApplicationRepository.java
package com.spring_reactive.reactive.repository;

import com.spring_reactive.reactive.model.JobApplication;
import org.springframework.data.mongodb.repository.ReactiveMongoRepository;
import org.springframework.stereotype.Repository;
import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

@Repository
public interface JobApplicationRepository extends
ReactiveMongoRepository<JobApplication, String> {
    Flux<JobApplication> findByJobId(String jobId);
    Flux<JobApplication> findByJobSeekerId(String jobSeekerId);
    Flux<JobApplication> findByStatus(String status);
    Mono<JobApplication> findByJobIdAndJobSeekerId(String jobId, String jobSeekerId);
}

/* ====================== SERVICES ====================== */
```

```
// src/main/java/com/spring_reactive/reactive/service/JobService.java
package com.spring_reactive.reactive.service;

import com.spring_reactive.reactive.model.Job;
import com.spring_reactive.reactive.repository.JobRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

import java.time.LocalDateTime;

@Service
public class JobService {
    private final JobRepository jobRepository;

    @Autowired
    public JobService(JobRepository jobRepository) {
        this.jobRepository = jobRepository;
    }

    public Flux<Job> getAllJobs() {
        return jobRepository.findAll();
    }

    public Flux<Job> getActiveJobs() {
```

```
    return jobRepository.findByActive(true);
}

public Mono<Job> getJobById(String id) {
    return jobRepository.findById(id);
}

public Mono<Job> createJob(Job job) {
    job.setPostedDate(LocalDateTime.now());
    return jobRepository.save(job);
}

public Mono<Job> updateJob(String id, Job job) {
    return jobRepository.findById(id)
            .flatMap(existingJob -> {
                existingJob.setTitle(job.getTitle());
                existingJob.setCompany(job.getCompany());
                existingJob.setDescription(job.getDescription());
                existingJob.setRequirements(job.getRequirements());
                existingJob.setLocation(job.getLocation());
                existingJob.setJobType(job.getJobType());
                existingJob.setSalary(job.getSalary());
                existingJob.setExpiryDate(job.getExpiryDate());
                existingJob.setActive(job.isActive());
                return jobRepository.save(existingJob);
            });
}
```

```
    public Mono<Void> deleteJob(String id) {
        return jobRepository.deleteById(id);
    }

    public Flux<Job> searchJobs(String keyword) {
        return
jobRepository.findByTitleContainingIgnoreCaseOrDescriptionContainingIgnoreCase(keyword,
keyword);
    }

    public Flux<Job> findJobsByLocation(String location) {
        return jobRepository.findByLocationContainingIgnoreCase(location);
    }

    public Flux<Job> findJobsByType(String type) {
        return jobRepository.findByJobType(type);
    }
}

// src/main/java/com/spring_reactive/reactive/service/JobSeekerService.java
package com.spring_reactive.reactive.service;

import com.spring_reactive.reactive.model.JobSeeker;
import com.spring_reactive.reactive.repository.JobSeekerRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import reactor.core.publisher.Flux;
```

```
import reactor.core.publisher.Mono;

import java.util.ArrayList;

@Service
public class JobSeekerService {

    private final JobSeekerRepository jobSeekerRepository;

    @Autowired
    public JobSeekerService(JobSeekerRepository jobSeekerRepository) {
        this.jobSeekerRepository = jobSeekerRepository;
    }

    public Flux<JobSeeker> getAllJobSeekers() {
        return jobSeekerRepository.findAll();
    }

    public Mono<JobSeeker> getJobSeekerById(String id) {
        return jobSeekerRepository.findById(id)
                .map(jobSeeker -> {
                    if (jobSeeker.getAppliedJobs() == null) {
                        jobSeeker.setAppliedJobs(new ArrayList<>());
                    }
                    return jobSeeker;
                });
    }
```

```
public Mono<JobSeeker> getJobSeekerByEmail(String email) {
    return jobSeekerRepository.findByEmail(email);
}

public Mono<JobSeeker> createJobSeeker(JobSeeker jobSeeker) {
    if (jobSeeker.getAppliedJobs() == null) {
        jobSeeker.setAppliedJobs(new ArrayList<>());
    }
    return jobSeekerRepository.save(jobSeeker);
}

public Mono<JobSeeker> updateJobSeeker(String id, JobSeeker jobSeeker) {
    return jobSeekerRepository.findById(id)
            .flatMap(existingJobSeeker -> {
                existingJobSeeker.setName(jobSeeker.getName());
                existingJobSeeker.setEmail(jobSeeker.getEmail());
                existingJobSeeker.setPhoneNumber(jobSeeker.getPhoneNumber());
                existingJobSeeker.setResumeUrl(jobSeeker.getResumeUrl());
                existingJobSeeker.setSkills(jobSeeker.getSkills());
                existingJobSeeker.setExperienceLevel(jobSeeker.getExperienceLevel());
                existingJobSeeker.setEducation(jobSeeker.getEducation());
                existingJobSeeker.setPreferredLocations(jobSeeker.getPreferredLocations());
                return jobSeekerRepository.save(existingJobSeeker);
            });
}
```

```
public Mono<Void> deleteJobSeeker(String id) {
    return jobSeekerRepository.deleteById(id);
}

public Flux<JobSeeker> findJobSeekersBySkill(String skill) {
    return jobSeekerRepository.findBySkillsContaining(skill);
}

public Flux<JobSeeker> findJobSeekersByExperienceLevel(String level) {
    return jobSeekerRepository.findByExperienceLevel(level);
}

public Flux<JobSeeker> findJobSeekersByLocation(String location) {
    return jobSeekerRepository.findByPreferredLocationsContaining(location);
}

public Mono<JobSeeker> addJobToAppliedList(String jobSeekerId, String jobId) {
    return jobSeekerRepository.findById(jobSeekerId)
            .flatMap(jobSeeker -> {
                if (jobSeeker.getAppliedJobs() == null) {
                    jobSeeker.setAppliedJobs(new ArrayList<>());
                }
                if (!jobSeeker.getAppliedJobs().contains(jobId)) {
                    jobSeeker.getAppliedJobs().add(jobId);
                }
                return jobSeekerRepository.save(jobSeeker);
            });
```

```
    }
}

// src/main/java/com/spring_reactive/reactive/service/JobApplicationService.java
package com.spring_reactive.reactive.service;

import com.spring_reactive.reactive.model.JobApplication;
import com.spring_reactive.reactive.repository.JobApplicationRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

@Service
public class JobApplicationService {

    private final JobApplicationRepository applicationRepository;
    private final JobSeekerService jobSeekerService;

    @Autowired
    public JobApplicationService(JobApplicationRepository applicationRepository,
JobSeekerService jobSeekerService) {
        this.applicationRepository = applicationRepository;
        this.jobSeekerService = jobSeekerService;
    }

    public Flux<JobApplication> getAllApplications() {
        return applicationRepository.findAll();
```

```
    }

    public Mono<JobApplication> getApplicationById(String id) {
        return applicationRepository.findById(id);
    }

    public Mono<JobApplication> saveApplication(JobApplication application) {
        // When a job application is created, update the jobSeeker's appliedJobs list
        return applicationRepository.save(application)
                .flatMap(savedApp ->
                    jobSeekerService.addJobToAppliedList(savedApp.getJobSeekerId(),
savedApp.getJobId())
                        .thenReturn(savedApp)
                );
    }

    public Mono<Void> deleteApplication(String id) {
        return applicationRepository.deleteById(id);
    }

    public Flux<JobApplication> getApplicationsByJobId(String jobId) {
        return applicationRepository.findByJobId(jobId);
    }

    public Flux<JobApplication> getApplicationsByJobSeekerId(String jobSeekerId) {
        return applicationRepository.findByJobSeekerId(jobSeekerId);
    }
```

```
    public Flux<JobApplication> getApplicationsByStatus(String status) {
        return applicationRepository.findByStatus(status);
    }

    public Mono<Boolean> hasApplied(String jobId, String jobSeekerId) {
        return applicationRepository.findByJobIdAndJobSeekerId(jobId, jobSeekerId)
                .map(app -> true)
                .defaultIfEmpty(false);
    }
}

/* ======================= CONTROLLERS ======================= */

// src/main/java/com/spring_reactive/reactive/controller/JobController.java
package com.spring_reactive.reactive.controller;

import com.spring_reactive.reactive.model.Job;
import com.spring_reactive.reactive.service.JobService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

@RestController
@RequestMapping("/api/jobs")
```

```
@CrossOrigin(origins = "*")
public class JobController {

    private final JobService jobService;

    @Autowired
    public JobController(JobService jobService) {
        this.jobService = jobService;
    }

    @GetMapping
    public Flux<Job> getAllJobs() {
        return jobService.getAllJobs();
    }

    @GetMapping("/active")
    public Flux<Job> getActiveJobs() {
        return jobService.getActiveJobs();
    }

    @GetMapping("/{id}")
    public Mono<ResponseEntity<Job>> getJobById(@PathVariable String id) {
        return jobService.getJobById(id)
                .map(job -> ResponseEntity.ok(job))
                .defaultIfEmpty(ResponseEntity.notFound().build());
    }
```

```
    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public Mono<Job> createJob(@RequestBody Job job) {
        return jobService.createJob(job);
    }

    @PutMapping("/{id}")
    public Mono<ResponseEntity<Job>> updateJob(@PathVariable String id, @RequestBody
Job job) {
        return jobService.updateJob(id, job)
                .map(updatedJob -> ResponseEntity.ok(updatedJob))
                .defaultIfEmpty(ResponseEntity.notFound().build());
    }

    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> deleteJob(@PathVariable String id) {
        return jobService.deleteJob(id);
    }

    @GetMapping("/search")
    public Flux<Job> searchJobs(@RequestParam String keyword) {
        return jobService.searchJobs(keyword);
    }

    @GetMapping("/location/{location}")
    public Flux<Job> findJobsByLocation(@PathVariable String location) {
        return jobService.findJobsByLocation(location);
```

```
    }

    @GetMapping("/type/{type}")
    public Flux<Job> findJobsByType(@PathVariable String type) {
        return jobService.findJobsByType(type);
    }
}

// src/main/java/com/spring_reactive/reactive/controller/JobSeekerController.java
package com.spring_reactive.reactive.controller;

import com.spring_reactive.reactive.model.JobSeeker;
import com.spring_reactive.reactive.service.JobSeekerService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

@RestController
@RequestMapping("/api/job-seekers")
@CrossOrigin(origins = "*")
public class JobSeekerController {

    private final JobSeekerService jobSeekerService;
```

```
    @Autowired
    public JobSeekerController(JobSeekerService jobSeekerService) {
        this.jobSeekerService = jobSeekerService;
    }

    @GetMapping
    public Flux<JobSeeker> getAllJobSeekers() {
        return jobSeekerService.getAllJobSeekers();
    }

    @GetMapping("/{id}")
    public Mono<ResponseEntity<JobSeeker>> getJobSeekerById(@PathVariable String id) {
        return jobSeekerService.getJobSeekerById(id)
                .map(jobSeeker -> ResponseEntity.ok(jobSeeker))
                .defaultIfEmpty(ResponseEntity.notFound().build());
    }

    @GetMapping("/email/{email}")
    public Mono<ResponseEntity<JobSeeker>> getJobSeekerByEmail(@PathVariable String
email) {
        return jobSeekerService.getJobSeekerByEmail(email)
                .map(jobSeeker -> ResponseEntity.ok(jobSeeker))
                .defaultIfEmpty(ResponseEntity.notFound().build());
    }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public Mono<JobSeeker> createJobSeeker(@RequestBody JobSeeker jobSeeker) {
```

```
        return jobSeekerService.createJobSeeker(jobSeeker);
    }

    @PutMapping("/{id}")
    public Mono<ResponseEntity<JobSeeker>> updateJobSeeker(@PathVariable String id,
@RequestBody JobSeeker jobSeeker) {
        return jobSeekerService.updateJobSeeker(id, jobSeeker)
                .map(updatedJobSeeker -> ResponseEntity.ok(updatedJobSeeker))
                .defaultIfEmpty(ResponseEntity.notFound().build());
    }

    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> deleteJobSeeker(@PathVariable String id) {
        return jobSeekerService.deleteJobSeeker(id);
    }

    @GetMapping("/skill/{skill}")
    public Flux<JobSeeker> findJobSeekersBySkill(@PathVariable String skill) {
        return jobSeekerService.findJobSeekersBySkill(skill);
    }

    @GetMapping("/level/{level}")
    public Flux<JobSeeker> findJobSeekersByExperienceLevel(@PathVariable String level) {
        return jobSeekerService.findJobSeekersByExperienceLevel(level);
    }

    @GetMapping("/location/{location}")
```

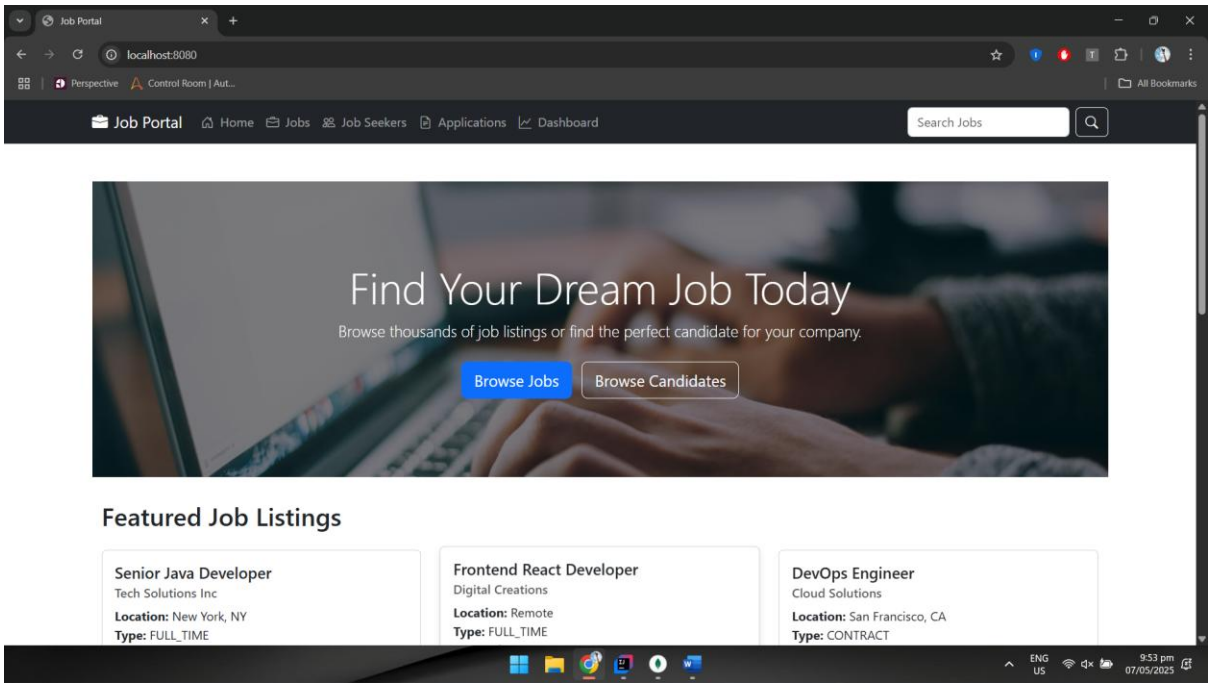
```
    public Flux<JobSeeker> findJobSeekersByLocation(@PathVariable String location) {
        return jobSeekerService.findJobSeekersByLocation(location);
    }
}

// src/main/java/com/spring_reactive/reactive/controller/JobApplicationController.java
package com.spring_reactive.reactive.controller;

import com.spring_reactive.reactive.model.JobApplication;
import com.spring_reactive.reactive.service.JobApplicationService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;

import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.time.LocalDateTime;

@RestController
@RequestMapping("/api/applications")
@CrossOrigin(origins = "*")
public class JobApplicationController {
```

```
    private final JobApplicationService applicationService;

    @Autowired
    public JobApplicationController(JobApplicationService applicationService) {
        this.applicationService = applicationService;
    }

    @GetMapping
    public Flux<JobApplication> getAllApplications() {
        return applicationService.getAllApplications();
    }

    @GetMapping("/{id}")
    public Mono<ResponseEntity<JobApplication>> getApplicationById(@PathVariable String
id) {
        return applicationService.getApplicationById(id)
                .map(application -> ResponseEntity.ok(application))
                .defaultIfEmpty(ResponseEntity.notFound().build());
    }

    @GetMapping("/job/{jobId}")
    public Flux<JobApplication> getApplicationsByJobId(@PathVariable String jobId) {
        return applicationService.getApplicationsByJobId(jobId);
    }

    @GetMapping("/job-seeker/{jobSeekerId}")
    public Flux<JobApplication> getApplicationsByJobSeekerId(@PathVariable String
jobSeekerId) {
```

```
        return applicationService.getApplicationsByJobSeekerId(jobSeekerId);
    }

    @PostMapping(consumes = MediaType.APPLICATION_FORM_URLENCODED_VALUE)
    public void createApplication(
            @RequestParam("jobId") String jobId,
            @RequestParam("jobSeekerId") String jobSeekerId,
            @RequestParam("coverLetter") String coverLetter,
            HttpServletResponse response) throws IOException {

        JobApplication application = new JobApplication(
                jobId,
                jobSeekerId,
                LocalDateTime.now(),
                "APPLIED",
                coverLetter
        );

        // Save the application (blocking for MVC)
        applicationService.saveApplication(application).block();

        // Redirect to applications page
        response.sendRedirect("/applications");
    }

    @PostMapping(path = "/{id}/status", consumes =
MediaType.APPLICATION_FORM_URLENCODED_VALUE)
    public void updateApplicationStatus(
```

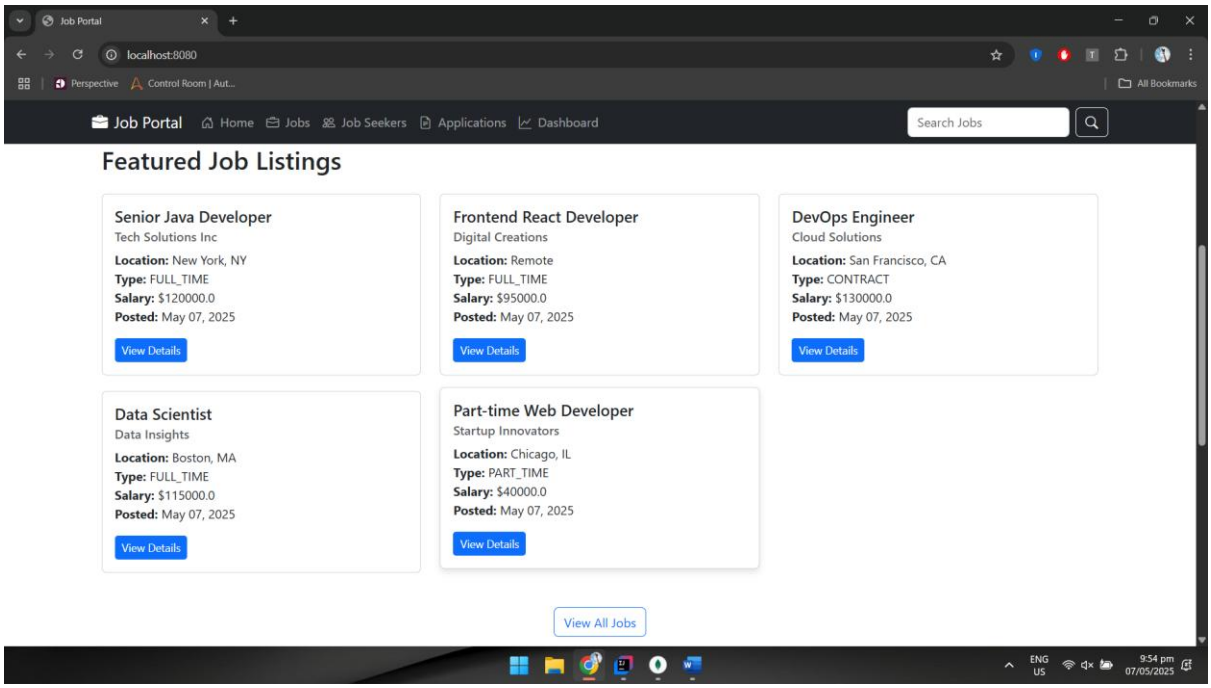
```
    @PathVariable String id,
    @RequestParam("status") String status,
    HttpServletResponse response) throws IOException {

JobApplication application = applicationService.getApplicationById(id).block();
if (application != null) {
    application.setStatus(status);
    applicationService.saveApplication(application).block();
}

// Redirect to applications
```

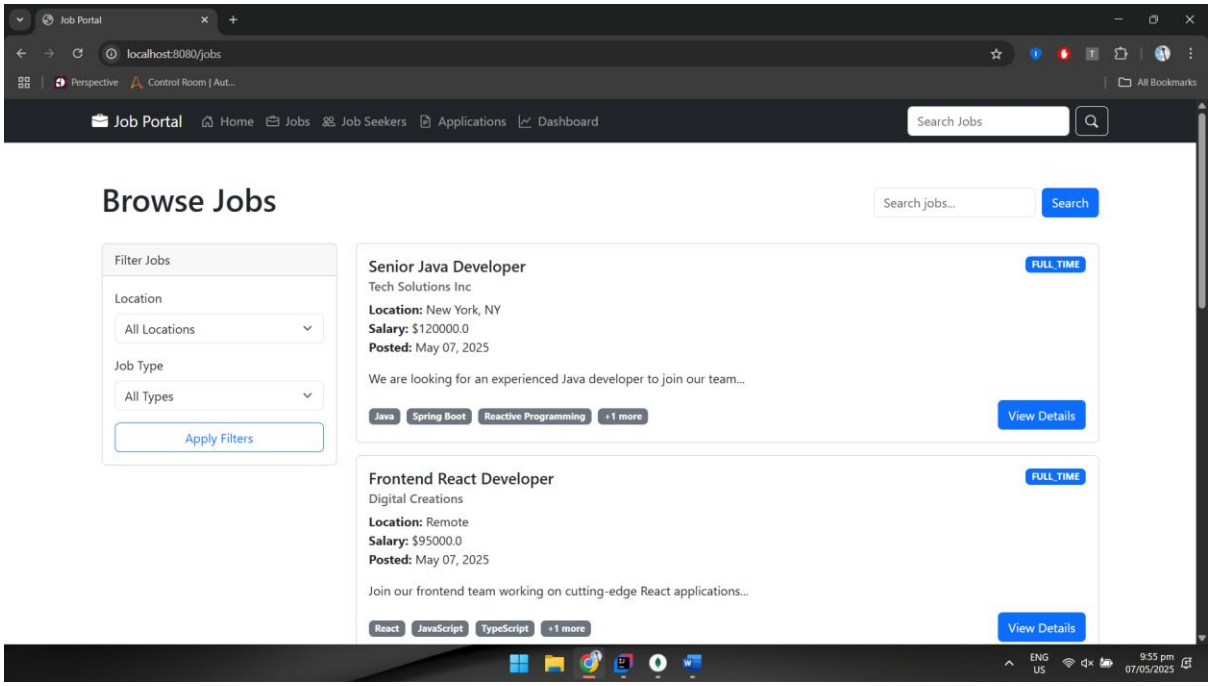
Output:



Academic Year: 2024-25 **Semester: II** **Class: FYMCA**
Course Code: MC506 **Course Name: Advanced Java Programming**

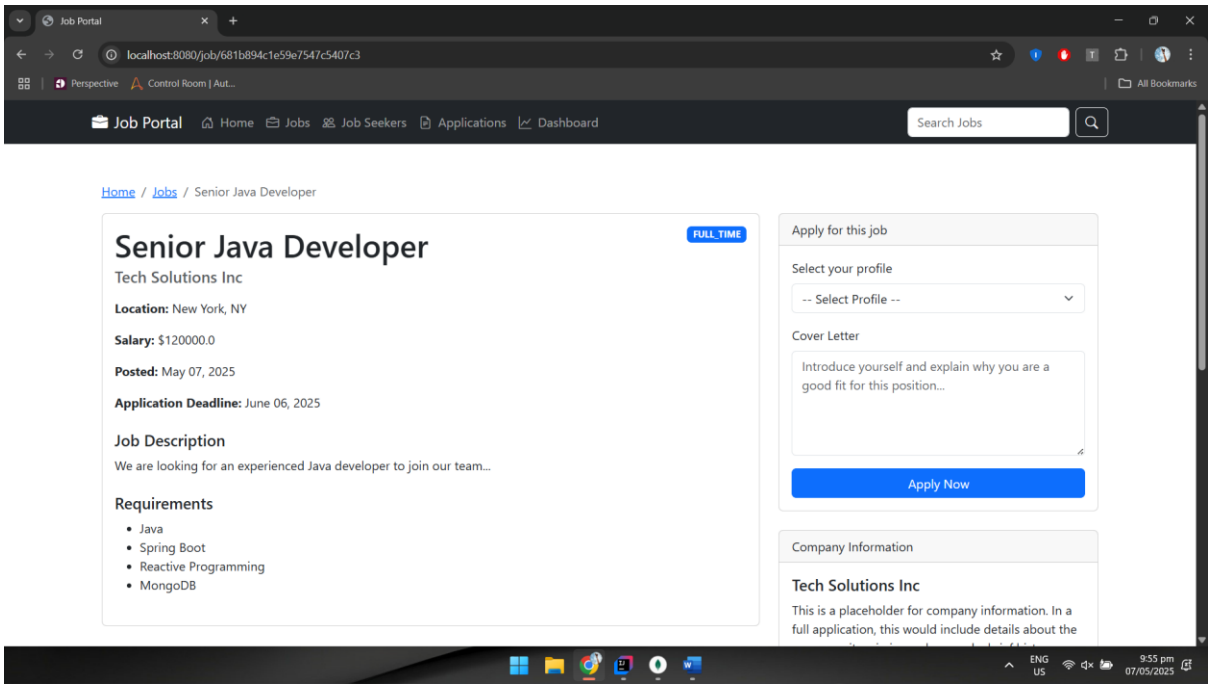
Atharva Vasant Angre **Practical 10** **2024510001**

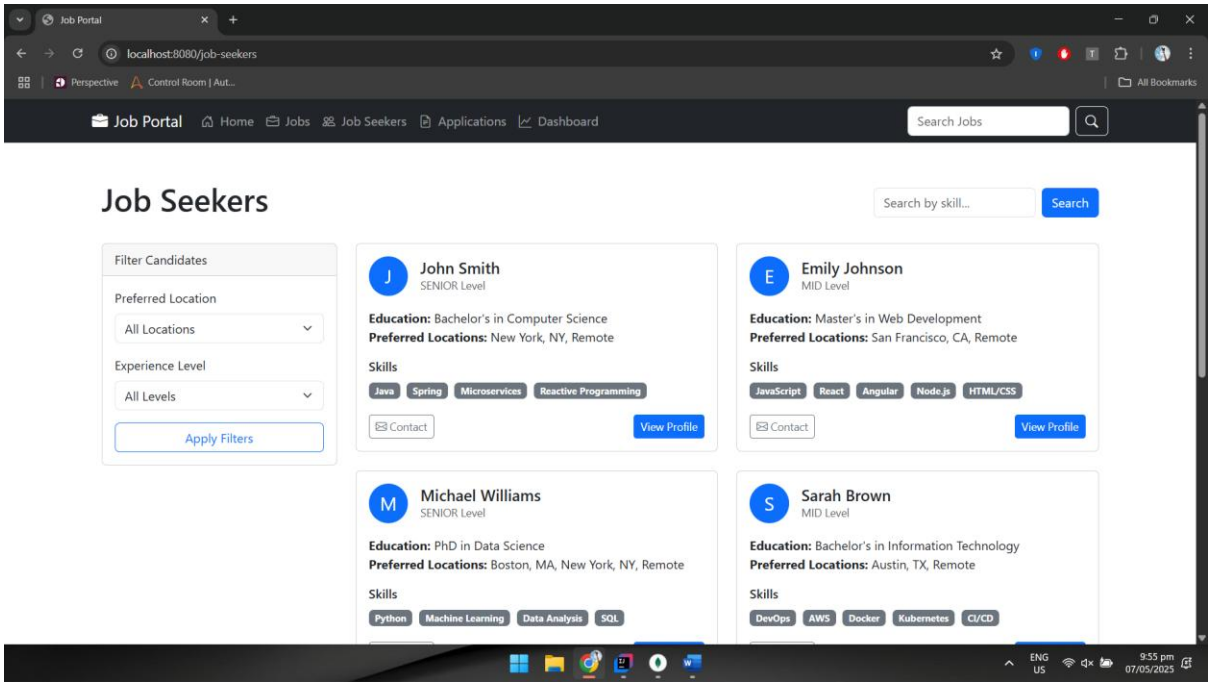




Academic Year: 2024-25 **Semester: II** **Class: FYMCA**
Course Code: MC506 **Course Name: Advanced Java Programming**

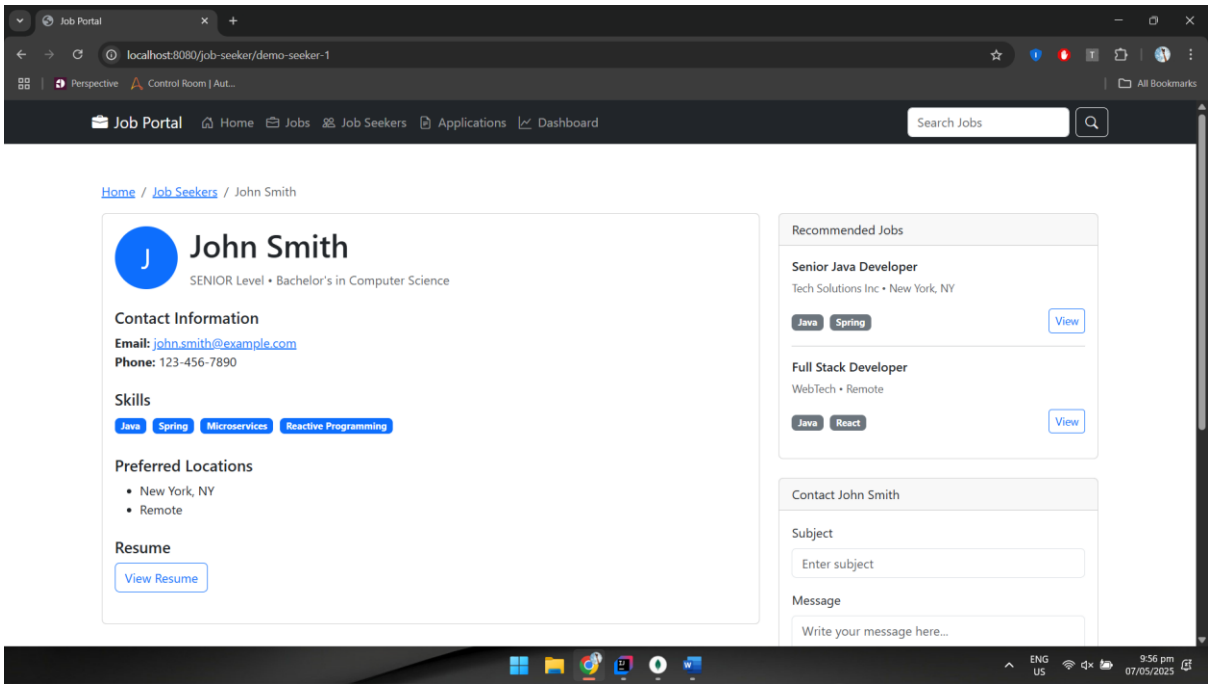
Atharva Vasant Angre **Practical 10** **2024510001**

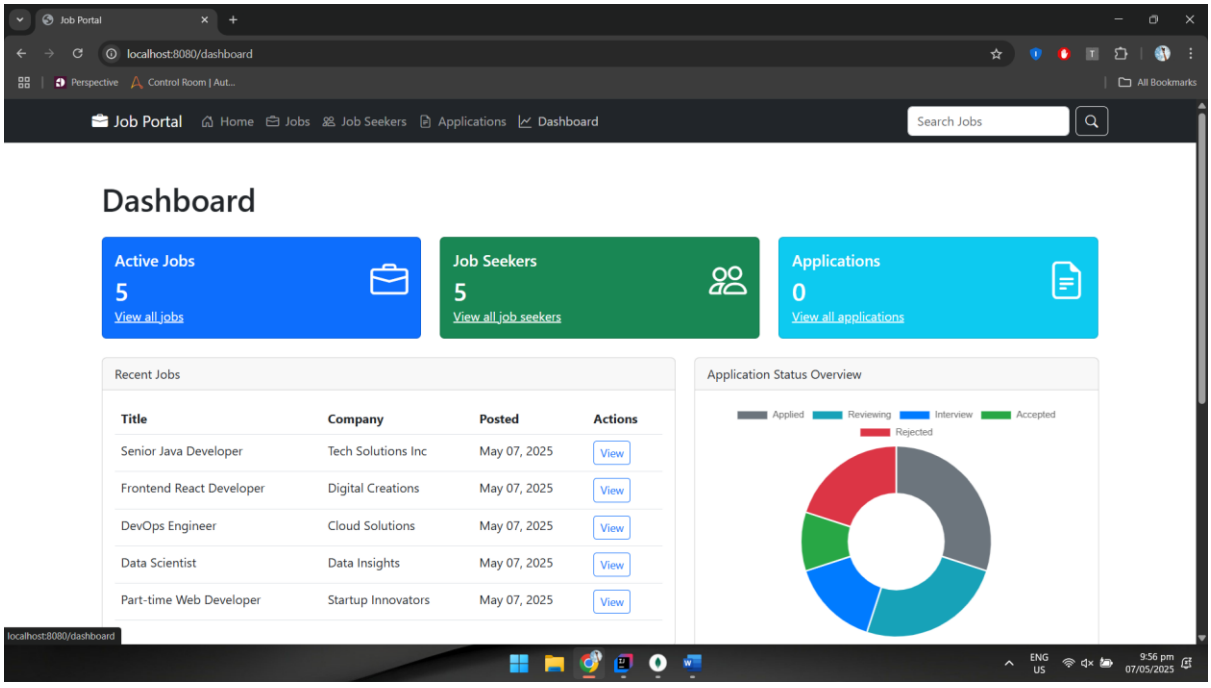




Academic Year: 2024-25 **Semester: II** **Class: FYMCA**
Course Code: MC506 **Course Name: Advanced Java Programming**

Atharva Vasant Angre **Practical 10** **2024510001**





Observation: This Spring Boot reactive job portal application successfully integrates WebFlux and Spring Data Reactive MongoDB to create a non-blocking, responsive job management platform. By leveraging reactive programming patterns and Flux/Mono data types throughout all layers of the application, it achieves high concurrency and scalability. The architecture follows a clean separation of concerns with domain models, repositories, services, and controllers working seamlessly together. Despite initial challenges with mixing WebFlux and MVC paradigms, the final solution demonstrates the power of reactive programming in building modern, efficient web applications that can handle high traffic volumes while maintaining optimal performance.