```
package com.internlink.internlink.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PatchMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.internlink.internlink.model.Application;
import com.internlink.internlink.service.ApplicationService;
import com.internlink.internlink.service.InteractionService;

@CrossOrigin(origins = "http://localhost:5173")
@RestController
@RequestMapping("/api/applications")
public class ApplicationController {

    @Autowired
    private ApplicationService applicationService;

    @Autowired

    private InteractionService interactionService;


    // Apply for an internship

    @PostMapping("/{internshipId}/apply")

    public ResponseEntity<String> apply(@PathVariable String internshipId,
@RequestParam String studentId) {

        try {

            // Call the service to save the application

            applicationService.saveApplication(internshipId, studentId);


            // Optionally record the interaction

            interactionService.saveInteraction(studentId, internshipId, "applied");


            return ResponseEntity.ok("Application submitted successfully!");

        } catch (Exception e) {

            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)

                    .body("An error occurred while processing the application");

        }

    }


    @GetMapping("/student/{studentId}/applications")

    public ResponseEntity<List<Application>>
getStudentApplications(@PathVariable String studentId) {

        try {

            // Fetch applications for the student
            List<Application> applications =
applicationService.findApplicationsByStudentId(studentId);

            // Return the applications
            return ResponseEntity.ok(applications);
        } catch (Exception e) {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
        }
    }

    @GetMapping("/internship/{internshipId}/applications")
    public ResponseEntity<List<Application>>
getApplicationsByInternship(@PathVariable String internshipId) {
        try {
            // Fetch applications for the specified internship
            List<Application> applications =
applicationService.findApplicationsByInternshipId(internshipId);

            // Return the applications
            return ResponseEntity.ok(applications);
        } catch (Exception e) {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
        }
    }

    @PatchMapping("/{applicationId}/update-status")
    public ResponseEntity<String> updateApplicationStatus(
            @PathVariable String applicationId,
            @RequestParam String status) {
        try {
            // Call the service to update the status
            applicationService.updateStatus(applicationId, status);

            return ResponseEntity.ok("Application status updated successfully!");
        } catch (Exception e) {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body("An error occurred while updating the application status");
        }

    }

}

package com.internlink.internlink.service;

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

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.data.mongodb.core.query.Criteria;
import org.springframework.data.mongodb.core.query.Query;
import org.springframework.data.mongodb.core.query.Update;
import org.springframework.stereotype.Service;

import com.internlink.internlink.model.Application;
import com.internlink.internlink.repository.ApplicationRepository;
import com.internlink.internlink.repository.StudentRepository;

@Service
public class ApplicationService {

    @Autowired
    private MongoTemplate mongoTemplate;
    @Autowired
    private ApplicationRepository applicationRepository;

    @Autowired
    private StudentRepository studentRepository; // Added repository to get student
details

    @Autowired
    private MessageService messageService; // Notification service

    public void saveApplication(String internshipId, String studentId) {
        // Create a new Application object
        Application application = new Application();
        application.setInternshipId(internshipId);
        application.setStudentId(studentId);
        application.setStatus("Pending"); // Set the status to "Pending"
        application.setAppliedOn(LocalDateTime.now()); // Set the application
timestamp


        // Save the Application to the MongoDB collection
        mongoTemplate.save(application, "applications"); // "applications" is the
collection name
    }


    public List<Application> findApplicationsByStudentId(String studentId) {
        Query query = new Query();
        query.addCriteria(Criteria.where("studentId").is(studentId));
        return mongoTemplate.find(query, Application.class, "applications");
    }


    public List<Application> findApplicationsByInternshipId(String internshipId) {
        Query query = new Query();
        query.addCriteria(Criteria.where("internshipId").is(internshipId));
        return mongoTemplate.find(query, Application.class, "applications");
    }


    public void updateStatus(String applicationId, String status) {
        // Validate that the status is one of the allowed values
        if (!status.equals("Accepted") && !status.equals("Rejected") &&
!status.equals("Pending")) {
            throw new IllegalArgumentException("Invalid status value");
        }
        // Build the query to find the application

        Query query = new Query();

        query.addCriteria(Criteria.where("id").is(applicationId));


        // Define the update operation

        Update update = new Update();

        update.set("status", status);


        // Perform the update

        mongoTemplate.updateFirst(query, update, Application.class);

    }


    public Application getApplicationById(String applicationId) {

        Query query = new Query();

        query.addCriteria(Criteria.where("id").is(applicationId));

        return mongoTemplate.findOne(query, Application.class, "applications");

    }



}


package com.internlink.internlink.model;


import java.time.LocalDateTime;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "interactions") // MongoDB collection name
public class Interaction {

    @Id
    private String id;
    private String studentId;
    private String internshipId;
    private String interactionType; // viewed, applied, accepted
    private int interactionScore;
    private LocalDateTime timestamp;

    // Constructor
    public Interaction(String studentId, String internshipId, String interactionType, int
interactionScore) {
        this.studentId = studentId;
        this.internshipId = internshipId;
        this.interactionType = interactionType;
        this.interactionScore = interactionScore;
        this.timestamp = LocalDateTime.now();
    }

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

    public String getStudentId() {
        return studentId;
    }

    public String getInternshipId() {
        return internshipId;
    }

    public String getInteractionType() {
        return interactionType;
    }

    public int getInteractionScore() {
        return interactionScore;
    }

    public LocalDateTime getTimestamp() {
        return timestamp;
    }

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

```
  public void setStudentId(String studentId) {
    this.studentId = studentId;
  }

  public void setInternshipId(String internshipId) {
    this.internshipId = internshipId;
  }

  public void setInteractionType(String interactionType) {
    this.interactionType = interactionType;
  }

  public void setInteractionScore(int interactionScore) {
    this.interactionScore = interactionScore;
  }

  public void setTimestamp(LocalDateTime timestamp) {
    this.timestamp = timestamp;
  }
}

package com.internlink.internlink.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.stereotype.Service;

import com.internlink.internlink.model.Interaction;

@Service
public class InteractionService {

    @Autowired
    private MongoTemplate mongoTemplate;

    public void saveInteraction(String studentId, String internshipId, String
interactionType) {
        int score = switch (interactionType) {
            case "applied" -> 3;
            case "accepted" -> 5;
            default -> 1; // Default is "viewed"
        };

        Interaction interaction = new Interaction(studentId, internshipId,
interactionType, score);
        mongoTemplate.save(interaction);
    }

}

package com.internlink.internlink.service;
```

```
import java.util.ArrayList;
import java.util.List;

import org.bson.Document;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.stereotype.Service;

import com.internlink.internlink.model.Internship;
import com.internlink.internlink.model.Student;

@Service
public class InternshipService {

  @Autowired
  private MongoTemplate mongoTemplate;

  public void createInternship(Internship internship) {
    mongoTemplate.save(internship);
  }

  public Internship getInternshipById(String internshipId) {
    return mongoTemplate.findById(internshipId, Internship.class);
  }

  public List<Float> getStudentEmbedding(String studentId) {
    // Log the student ID being queried
    System.out.println("Fetching embedding for student ID: " + studentId);

    // Fetch the student from the database
    Student student = mongoTemplate.findById(studentId, Student.class);
    if (student == null) {
      // Log if the student is not found
      System.err.println("Student not found with ID: " + studentId);
      throw new IllegalStateException("Student embedding not found!");
    }

    // Check if the embedding exists
    List<Float> embedding = student.getEmbedding();
    if (embedding == null || embedding.isEmpty()) {
      // Log if the embedding is missing or empty
      System.err.println("Embedding not found or empty for student ID: " +
studentId);
      throw new IllegalStateException("Student embedding not found!");
    }

    // Log successful retrieval of the embedding
    System.out.println("Embedding successfully fetched for student ID: " +
studentId);
    return embedding;
  }
  public List<Internship> getRecommendedInternships(List<Float>
studentEmbedding) {
    System.out.println("Generating recommendations using student embedding: " +
studentEmbedding);

    if (studentEmbedding == null || studentEmbedding.isEmpty()) {
      System.err.println("Invalid student embedding: " + studentEmbedding);
      throw new IllegalArgumentException("Invalid student embedding!");
    }

    // ✅ Corrected: Add "limit" inside $vectorSearch
    Document vectorSearchQuery = new Document("$vectorSearch",
        new Document("index", "internship_index2")
            .append("queryVector", studentEmbedding)
            .append("path", "embedding")
            .append("numCandidates", 10) // Number of candidates considered
before ranking
            .append("k", 5) // Top K results to return
            .append("limit", 5) // ✅ FIX: "limit" inside $vectorSearch
    );

    // Add a $project stage to exclude the "embedding" field
    Document projectStage = new Document("$project",
        new Document("embedding", 0) // Exclude the "embedding" field
    );

    System.out.println("Vector search query: " + vectorSearchQuery.toJson());
```

```
        // Run aggregation with the $project stage
        List<Document> results = mongoTemplate.getCollection("internships")
            .aggregate(List.of(vectorSearchQuery, projectStage)) // Add $project stage
            .into(new ArrayList<>());

        // Convert BSON documents to Internship objects
        List<Internship> internships = results.stream()
            .map(doc -> mongoTemplate.getConverter().read(Internship.class, doc))
            .toList();

        System.out.println("Number of internships found: " + internships.size());
        return internships;
    }
}

package com.internlink.internlink.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PatchMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.internlink.internlink.model.Application;
import com.internlink.internlink.model.Internship;
import com.internlink.internlink.service.ApplicationService;
import com.internlink.internlink.service.EmbeddingService;
import com.internlink.internlink.service.InteractionService;
import com.internlink.internlink.service.InternshipService;

@RestController
@RequestMapping("/api/internships")
public class InternshipController {
  @Autowired
  private ApplicationService applicationService;
  @Autowired
  private InteractionService interactionService;

  @Autowired
  private InternshipService internshipService;

  @Autowired
  private EmbeddingService embeddingService;
```

```
    @PostMapping("/create")
    public ResponseEntity<String> createInternship(@RequestBody Internship
internship) {
        try {
            String text = internship.getDescription() + " " + String.join(" ",
internship.getRequiredSkills());
            List<Float> embedding = embeddingService.generateEmbedding(text);
            internship.setEmbedding(embedding);
            internshipService.createInternship(internship);
            return ResponseEntity.ok("Internship created successfully!");
        } catch (Exception e) {
            return ResponseEntity.internalServerError().body("Error creating internship: " +
e.getMessage());
        }
    }

    @GetMapping("/recommend")
    public ResponseEntity<List<Internship>>
recommendInternships(@RequestParam String studentId) {
        try {
            // Log the student ID being requested
            System.out.println("Fetching recommendations for student ID: " + studentId);

            // Fetch the student's embedding
            List<Float> studentEmbedding =
internshipService.getStudentEmbedding(studentId);
```

```
            System.out
                    .println("Student embedding fetched: " + (studentEmbedding != null ?
"Success" : "Null or empty"));

            // Get recommended internships
            List<Internship> recommendedInternships =
internshipService.getRecommendedInternships(studentEmbedding);
            System.out.println("Number of recommended internships: "
                    + (recommendedInternships != null ? recommendedInternships.size() :
"Null"));

            // Return the recommendations
            return ResponseEntity.ok(recommendedInternships);
        } catch (Exception e) {
            // Log the exception
            System.err.println("Error in recommendInternships: " + e.getMessage());
            e.printStackTrace();
            return ResponseEntity.internalServerError().build();
        }
    }

    @PatchMapping("/{applicationId}/update-status")
    public ResponseEntity<String> updateApplicationStatus(
            @PathVariable String applicationId,
            @RequestParam String status) {
        try {
            // Update the application status
            applicationService.updateStatus(applicationId, status);


            // If the status is "Accepted," log the interaction
            if ("Accepted".equalsIgnoreCase(status)) {
                // Fetch application details to get studentId and internshipId
                Application application =
applicationService.getApplicationById(applicationId);
                interactionService.saveInteraction(application.getStudentId(),
application.getInternshipId(),
                        "accepted");
            }


            return ResponseEntity.ok("Application status updated successfully!");
        } catch (Exception e) {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body("An error occurred while updating the application status");
        }
    }


}
```