Codes(Reviewer Entity)

1. **ReviewerApplication.java (Main File):**

```
package com.example.Reviewers;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.CommandLineRunner;
import org.springframework.context.annotation.Bean;

import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.SQLException;

@SpringBootApplication
public class ReviewerApplication {

  public static void main(String[] args) {
    SpringApplication.run(ReviewerApplication.class, args);
  }

  @Bean
  public CommandLineRunner testDatabaseConnection(DataSource dataSource) {
    return args -> {
      try (Connection connection = dataSource.getConnection()) {
        System.out.println("Database connection successful!");
        System.out.println("Connection: " + connection.getMetaData().getURL());
      } catch (SQLException e) {
        System.err.println("Database connection failed!");
        System.err.println("Error message: " + e.getMessage());
        System.err.println("SQL State: " + e.getSQLState());
        System.err.println("Error Code: " + e.getErrorCode());
        e.printStackTrace();
      }
    };
  }
}
```

2. **ReviewerEntity.java:**

```
package com.example.Reviewers.Entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "reviewers")
```

```
public class ReviewerEntity {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  private String name;
  private String email;
  private String affiliation;
  private String expertise;

  // Default constructor
  public ReviewerEntity() {
  }

  // Parameterized constructor
  public ReviewerEntity(int id, String name, String email, String affiliation, String expertise) {
    this.id = id;
    this.name = name;
    this.email = email;
    this.affiliation = affiliation;
    this.expertise = expertise;
  }

  // Getters
  public int getId() {
    return id;
  }

  public String getName() {
    return name;
  }

  public String getEmail() {
    return email;
  }

  public String getAffiliation() {
    return affiliation;
  }

  public String getExpertise() {
    return expertise;
  }

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

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

  public void setAffiliation(String affiliation) {
    this.affiliation = affiliation;
  }
```

```java
    public void setExpertise(String expertise) {
        this.expertise = expertise;
    }

    @Override
    public String toString() {
        return "Reviewer [id=" + id +
               ", name=" + name +
               ", email=" + email +
               ", affiliation=" + affiliation +
               ", expertise=" + expertise + "]";
    }
}
```

3. **ReviewerController.java:**

```java
package com.example.Reviewers.Controller;

import java.util.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
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.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.example.Reviewers.Entity.ReviewerEntity;
import com.example.Reviewers.Repository.ReviewerRepository;

import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.responses.ApiResponse;
import io.swagger.v3.oas.annotations.responses.ApiResponses;
import io.swagger.v3.oas.annotations.tags.Tag;

@RestController
@Tag(name = "Reviewer Management", description = "Operations related to reviewers")
public class ReviewerController {
    @Autowired
    private ReviewerRepository reviewerRepository;

    @GetMapping("/test")
    @Operation(summary = "Test Method", description = "Returns a greeting message.")
    public String getMethodName() {
        return "Hello Sachin";
    }

    @GetMapping("/reviewers")
    @Operation(summary = "Get All Reviewers", description = "Retrieves all reviewers.")
    public List<ReviewerEntity> getAllReviewers() {
        return reviewerRepository.findAllByOrderByIdAsc();
    }

    @GetMapping("/reviewers/{id}")
    @Operation(summary = "Get Reviewer by ID", description = "Retrieves a reviewer by their ID.")
    @ApiResponses(value = {
        @ApiResponse(responseCode = "200", description = "Successful retrieval"),
        @ApiResponse(responseCode = "404", description = "Reviewer not found")
    })
    public Optional<ReviewerEntity> getReviewerById(@PathVariable("id") Integer id) {
        return reviewerRepository.findById(id);
    }

    @PostMapping("/reviewers")
    @Operation(summary = "Create Reviewer", description = "Creates a new reviewer.")
    public ReviewerEntity createReviewer(@RequestBody ReviewerEntity reviewer) {
        return reviewerRepository.save(reviewer);
    }

    @PutMapping("/reviewers")
    @Operation(summary = "Update Reviewer", description = "Updates an existing reviewer.")
    public ReviewerEntity updateReviewer(@RequestBody ReviewerEntity reviewer) {
        return reviewerRepository.save(reviewer);
    }

    @PutMapping("/reviewers/{id}")
    @Operation(summary = "Full Update Reviewer", description = "Replaces an existing reviewer by their ID.")
    public ReviewerEntity putMethodName(@PathVariable("id") int id, @RequestBody ReviewerEntity reviewer)
{
        Optional<ReviewerEntity> temOptional = reviewerRepository.findById(id);
        if (temOptional.isPresent()) {
            return reviewerRepository.save(reviewer);
        } else {
            return reviewerRepository.save(reviewer);
        }
    }

    @PatchMapping("/reviewers/{id}")
    @Operation(summary = "Partial Update Reviewer", description = "Updates specific fields of an existing
reviewer.")
    public ReviewerEntity patchReviewer(@PathVariable("id") int id, @RequestBody Map<String, Object>
updates) {
        Optional<ReviewerEntity> existingReviewerOptional = reviewerRepository.findById(id);
        if (existingReviewerOptional.isPresent()) {
            ReviewerEntity existingReviewer = existingReviewerOptional.get();
            updates.forEach((key, value) -> {
                switch (key) {
                    case "name":
                        existingReviewer.setName((String) value);
                        break;
                    case "email":
                        existingReviewer.setEmail((String) value);
                        break;
                    case "affiliation":
                        existingReviewer.setAffiliation((String) value);
                        break;
                    case "expertise":
                        existingReviewer.setExpertise((String) value);
                        break;
                }
            });
            return reviewerRepository.save(existingReviewer);
        } else {
            throw new RuntimeException("Reviewer with id " + id + " not found");
        }
    }

    @DeleteMapping("/reviewers/{id}")
    @Operation(summary = "Delete Reviewer by ID", description = "Deletes a reviewer by their ID.")
    public String deleteReviewerById(@PathVariable("id") int id) {
        reviewerRepository.deleteById(id);
        return "deleted";
    }
}
```

4. **ReviewerRepository.java:**

```
package com.example.Reviewers.Repository;

import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import com.example.Reviewers.Entity.ReviewerEntity;

public interface ReviewerRepository extends JpaRepository<ReviewerEntity, Integer> {
    List<ReviewerEntity> findAllByOrderByIdAsc();
}
```