Auto Assign is not working ->

```
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.lawcubator.falcon.infra.model;

public enum WorkspaceUserRole {
   WORKSPACE_ADMIN,
   BRANCH_ADMIN,
   EMPLOYER,
   EMPLOYEE,
   LAWCUBATOR_ADMIN,
   BOT;

   private WorkspaceUserRole() {
   }
}
```

```
package com.lawcubator.falcon.lms.dtos.request.lms;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.util.List;

@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
public class AutoAssignBranchDepartmentRequest {
   private boolean isAllBranch;
   private List<BranchInfoRequest> branches;
}
```

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//
```

```java
package com.lawcubator.falcon.infra.workspace.dto;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.lawcubator.falcon.infra.model.WorkspaceUserRole;
import java.util.List;
import java.util.UUID;

@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
public class WorkspaceContext {
   private String userId;
   private UUID workspaceId;
   private String workspaceUrl;
   private String companyName;
   private UUID branchId;
   private List<WorkspaceUserRole> roles;

   public String toString() {
       return this.roles + " - WorkspaceContext{userId='" + this.userId + "',
workspaceId=" + this.workspaceId + ", workspaceUrl='" + this.workspaceUrl + "',
companyName='" + this.companyName + "', branchId=" + this.branchId + "}";
   }

   public boolean equals(final Object o) {
       if (o == this) {
           return true;
       } else if (!(o instanceof WorkspaceContext)) {
           return false;
       } else {
           WorkspaceContext other = (WorkspaceContext)o;
           if (!other.canEqual(this)) {
               return false;
           } else {
               Object this$userId = this.getUserId();
               Object other$userId = other.getUserId();
               if (this$userId == null) {
                   if (other$userId != null) {
                       return false;
                   }
               } else if (!this$userId.equals(other$userId)) {
                   return false;
               }

               Object this$workspaceId = this.getWorkspaceId();
               Object other$workspaceId = other.getWorkspaceId();
               if (this$workspaceId == null) {
                   if (other$workspaceId != null) {
```

```
            return false;
        }
    } else if (!this$workspaceId.equals(other$workspaceId)) {
        return false;
    }

    Object this$workspaceUrl = this.getWorkspaceUrl();
    Object other$workspaceUrl = other.getWorkspaceUrl();
    if (this$workspaceUrl == null) {
        if (other$workspaceUrl != null) {
            return false;
        }
    } else if (!this$workspaceUrl.equals(other$workspaceUrl)) {
        return false;
    }

    Object this$companyName = this.getCompanyName();
    Object other$companyName = other.getCompanyName();
    if (this$companyName == null) {
        if (other$companyName != null) {
            return false;
        }
    } else if (!this$companyName.equals(other$companyName)) {
        return false;
    }

    Object this$branchId = this.getBranchId();
    Object other$branchId = other.getBranchId();
    if (this$branchId == null) {
        if (other$branchId != null) {
            return false;
        }
    } else if (!this$branchId.equals(other$branchId)) {
        return false;
    }

    Object this$roles = this.getRoles();
    Object other$roles = other.getRoles();
    if (this$roles == null) {
        if (other$roles != null) {
            return false;
        }
    } else if (!this$roles.equals(other$roles)) {
        return false;
    }

    return true;
```

```
            }
        }
    }

    protected boolean canEqual(final Object other) {
        return other instanceof WorkspaceContext;
    }

    public int hashCode() {
        int PRIME = 59;
        int result = 1;
        Object $userId = this.getUserId();
        result = result * 59 + ($userId == null ? 43 : $userId.hashCode());
        Object $workspaceId = this.getWorkspaceId();
        result = result * 59 + ($workspaceId == null ? 43 :
$workspaceId.hashCode());
        Object $workspaceUrl = this.getWorkspaceUrl();
        result = result * 59 + ($workspaceUrl == null ? 43 :
$workspaceUrl.hashCode());
        Object $companyName = this.getCompanyName();
        result = result * 59 + ($companyName == null ? 43 :
$companyName.hashCode());
        Object $branchId = this.getBranchId();
        result = result * 59 + ($branchId == null ? 43 : $branchId.hashCode());
        Object $roles = this.getRoles();
        result = result * 59 + ($roles == null ? 43 : $roles.hashCode());
        return result;
    }

    public WorkspaceContext(final String userId, final UUID workspaceId, final
String workspaceUrl, final String companyName, final UUID branchId, final
List<WorkspaceUserRole> roles) {
        this.userId = userId;
        this.workspaceId = workspaceId;
        this.workspaceUrl = workspaceUrl;
        this.companyName = companyName;
        this.branchId = branchId;
        this.roles = roles;
    }

    public WorkspaceContext() {
    }

    public String getUserId() {
        return this.userId;
    }
```

```
    public UUID getWorkspaceId() {
        return this.workspaceId;
    }

    public String getWorkspaceUrl() {
        return this.workspaceUrl;
    }

    public String getCompanyName() {
        return this.companyName;
    }

    public UUID getBranchId() {
        return this.branchId;
    }

    public List<WorkspaceUserRole> getRoles() {
        return this.roles;
    }

    public void setUserId(final String userId) {
        this.userId = userId;
    }

    public void setWorkspaceId(final UUID workspaceId) {
        this.workspaceId = workspaceId;
    }

    public void setWorkspaceUrl(final String workspaceUrl) {
        this.workspaceUrl = workspaceUrl;
    }

    public void setCompanyName(final String companyName) {
        this.companyName = companyName;
    }

    public void setBranchId(final UUID branchId) {
        this.branchId = branchId;
    }

    public void setRoles(final List<WorkspaceUserRole> roles) {
        this.roles = roles;
    }
}
```

```
package com.lawcubator.falcon.lms.model.db.compliancepolicy;
```

```java
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.lawcubator.falcon.infra.annotations.DTOIgnore;
import com.lawcubator.falcon.infra.annotations.EntityOwners;
import com.lawcubator.falcon.infra.annotations.Validate;
import com.lawcubator.falcon.infra.model.Model;
import com.lawcubator.falcon.infra.model.WorkspaceUserRole;
import com.lawcubator.falcon.lms.constants.ContentStatus;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.EntityListeners;
import jakarta.persistence.EnumType;
import jakarta.persistence.Enumerated;
import jakarta.persistence.OneToMany;
import lombok.Data;
import lombok.EqualsAndHashCode;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import java.util.List;

@EqualsAndHashCode(callSuper = true)
@Entity(name = "policy")
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
@EntityListeners(AuditingEntityListener.class)
@Data
@EntityOwners(roles = {WorkspaceUserRole.BRANCH_ADMIN,
WorkspaceUserRole.WORKSPACE_ADMIN, WorkspaceUserRole.EMPLOYER,
   WorkspaceUserRole.LAWCUBATOR_ADMIN})
public class Policy extends Model {
   @Column(nullable = false)
   @Validate(nonNullable = true, nonEmpty = true, displayName = "Policy Name")
   private String policyName;

   @Column(nullable = false)
   private boolean isLawcubatorOwned;

   @Column(columnDefinition = "text")
   @Validate(displayName = "Policy Description")
   private String policyDescription;

   @Validate(displayName = "Policy bucket name")
   private String policyBucketName;

   @Column(columnDefinition = "text")
   @Validate(displayName = "Policy PDF object key")
   private String policyPdfObjectKey;
```

```
   @Column(columnDefinition = "text")
   @Validate(displayName = "Policy DOCx object key")
   private String policyDocxObjectKey;

   @Column(columnDefinition = "text")
   @Validate(displayName = "Policy PDF url")
   private String policyPdfUrl;

   @Column(columnDefinition = "text")
   @Validate(displayName = "Policy DOCx url")
   private String policyDocxUrl;

   private boolean pdfUploadConfirmed = false;

   private boolean docxUploadConfirmed = false;

   @DTOIgnore
   @OneToMany(mappedBy = "policy", cascade = CascadeType.REMOVE)
   private List<PolicyFaqs> policyFaqs;

   @DTOIgnore
   @OneToMany(mappedBy = "policy", cascade = CascadeType.REMOVE)
   private List<WorkspacePolicy> workspacePolicies;

   @OneToMany(mappedBy = "policy", cascade = CascadeType.REMOVE)
   private List<PolicyCompliance> policyCompliances;

   @Validate(nonEmpty = true, nonNullable = true, displayName = "Policy
status")
   @Enumerated(EnumType.STRING)
   private ContentStatus policyStatus;
}
```

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.lawcubator.falcon.infra.model;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import com.lawcubator.falcon.infra.annotations.EntityOwners;
import com.lawcubator.falcon.infra.core.InstantJsonDeSerializer;
import com.lawcubator.falcon.infra.core.InstantJsonSerializer;
```

```java
import com.lawcubator.falcon.infra.enums.RecordStatus;
import jakarta.persistence.Column;
import jakarta.persistence.EnumType;
import jakarta.persistence.Enumerated;
import jakarta.persistence.Id;
import jakarta.persistence.MappedSuperclass;
import jakarta.persistence.PrePersist;
import java.time.Instant;
import java.util.UUID;
import org.springframework.data.annotation.CreatedBy;
import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.annotation.LastModifiedBy;
import org.springframework.data.annotation.LastModifiedDate;

@MappedSuperclass
@EntityOwners(
   roles = {WorkspaceUserRole.WORKSPACE_ADMIN}
)
public class Model implements IModel {
   @Id
   private UUID id;
   @CreatedBy
   private String createdBy;
   @LastModifiedBy
   private String updatedBy;
   @CreatedDate
   @JsonDeserialize(
       using = InstantJsonDeSerializer.class
   )
   @JsonSerialize(
       using = InstantJsonSerializer.class
   )
   private Instant createdDate;
   @LastModifiedDate
   @JsonDeserialize(
       using = InstantJsonDeSerializer.class
   )
   @JsonSerialize(
       using = InstantJsonSerializer.class
   )
   private Instant lastModifiedDate;
   @Enumerated(EnumType.STRING)
   private RecordStatus status;
   @Column
   private UUID workspaceId;
   @Column
   private UUID branchId;
```

```
    @PrePersist
    private void ensureId() {
        this.setId(UUID.randomUUID());
    }

    public Model() {
        this.status = RecordStatus.ACTIVE;
    }

    public Model(final UUID id, final String createdBy, final String updatedBy,
final Instant createdDate, final Instant lastModifiedDate, final RecordStatus
status, final UUID workspaceId, final UUID branchId) {
        this.status = RecordStatus.ACTIVE;
        this.id = id;
        this.createdBy = createdBy;
        this.updatedBy = updatedBy;
        this.createdDate = createdDate;
        this.lastModifiedDate = lastModifiedDate;
        this.status = status;
        this.workspaceId = workspaceId;
        this.branchId = branchId;
    }

    public UUID getId() {
        return this.id;
    }

    public String getCreatedBy() {
        return this.createdBy;
    }

    public String getUpdatedBy() {
        return this.updatedBy;
    }

    public Instant getCreatedDate() {
        return this.createdDate;
    }

    public Instant getLastModifiedDate() {
        return this.lastModifiedDate;
    }

    public RecordStatus getStatus() {
        return this.status;
    }
```

```
    public UUID getWorkspaceId() {
        return this.workspaceId;
    }

    public UUID getBranchId() {
        return this.branchId;
    }

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

    public void setCreatedBy(final String createdBy) {
        this.createdBy = createdBy;
    }

    public void setUpdatedBy(final String updatedBy) {
        this.updatedBy = updatedBy;
    }

    @JsonDeserialize(
        using = InstantJsonDeSerializer.class
    )
    public void setCreatedDate(final Instant createdDate) {
        this.createdDate = createdDate;
    }

    @JsonDeserialize(
        using = InstantJsonDeSerializer.class
    )
    public void setLastModifiedDate(final Instant lastModifiedDate) {
        this.lastModifiedDate = lastModifiedDate;
    }

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

    public void setWorkspaceId(final UUID workspaceId) {
        this.workspaceId = workspaceId;
    }

    public void setBranchId(final UUID branchId) {
        this.branchId = branchId;
    }
}
```

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.lawcubator.falcon.infra.model;

import java.util.UUID;

public interface IModel {
   UUID getWorkspaceId();

   UUID getBranchId();
}
```

```
package com.lawcubator.falcon.lms.model.db.compliancepolicy;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.lawcubator.falcon.infra.annotations.EntityOwners;
import com.lawcubator.falcon.infra.model.Model;
import com.lawcubator.falcon.infra.model.WorkspaceUserRole;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.EntityListeners;
import jakarta.persistence.ManyToOne;
import lombok.Data;
import lombok.EqualsAndHashCode;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import java.time.Instant;
import java.util.UUID;

@EqualsAndHashCode(callSuper = true)
@Entity(name = "workspace_policy_assign_info")
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
@EntityListeners(AuditingEntityListener.class)
@Data
@EntityOwners(roles = {WorkspaceUserRole.BRANCH_ADMIN,
WorkspaceUserRole.WORKSPACE_ADMIN})
public class WorkspacePolicyAssignInfo extends Model {
   @Column(nullable = true)
   private UUID departmentId;

   @ManyToOne
   private WorkspacePolicy workspacePolicy;
```

```
    private Instant assignedOn;

    private UUID assignedBy;

    @Column
    private boolean isForAllBranches;

}
```

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.lawcubator.falcon.infra.model;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import com.lawcubator.falcon.infra.annotations.EntityOwners;
import com.lawcubator.falcon.infra.core.InstantJsonDeSerializer;
import com.lawcubator.falcon.infra.core.InstantJsonSerializer;
import com.lawcubator.falcon.infra.enums.RecordStatus;
import jakarta.persistence.Column;
import jakarta.persistence.EnumType;
import jakarta.persistence.Enumerated;
import jakarta.persistence.Id;
import jakarta.persistence.MappedSuperclass;
import jakarta.persistence.PrePersist;
import java.time.Instant;
import java.util.UUID;
import org.springframework.data.annotation.CreatedBy;
import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.annotation.LastModifiedBy;
import org.springframework.data.annotation.LastModifiedDate;

@MappedSuperclass
@EntityOwners(
    roles = {WorkspaceUserRole.WORKSPACE_ADMIN}
)
public class Model implements IModel {
    @Id
    private UUID id;
    @CreatedBy
    private String createdBy;
    @LastModifiedBy
```

```
    private String updatedBy;
    @CreatedDate
    @JsonDeserialize(
        using = InstantJsonDeSerializer.class
    )
    @JsonSerialize(
        using = InstantJsonSerializer.class
    )
    private Instant createdDate;
    @LastModifiedDate
    @JsonDeserialize(
        using = InstantJsonDeSerializer.class
    )
    @JsonSerialize(
        using = InstantJsonSerializer.class
    )
    private Instant lastModifiedDate;
    @Enumerated(EnumType.STRING)
    private RecordStatus status;
    @Column
    private UUID workspaceId;
    @Column
    private UUID branchId;

    @PrePersist
    private void ensureId() {
        this.setId(UUID.randomUUID());
    }

    public Model() {
        this.status = RecordStatus.ACTIVE;
    }

    public Model(final UUID id, final String createdBy, final String updatedBy,
final Instant createdDate, final Instant lastModifiedDate, final RecordStatus
status, final UUID workspaceId, final UUID branchId) {
        this.status = RecordStatus.ACTIVE;
        this.id = id;
        this.createdBy = createdBy;
        this.updatedBy = updatedBy;
        this.createdDate = createdDate;
        this.lastModifiedDate = lastModifiedDate;
        this.status = status;
        this.workspaceId = workspaceId;
        this.branchId = branchId;
    }
```

```
public UUID getId() {
    return this.id;
}

public String getCreatedBy() {
    return this.createdBy;
}

public String getUpdatedBy() {
    return this.updatedBy;
}

public Instant getCreatedDate() {
    return this.createdDate;
}

public Instant getLastModifiedDate() {
    return this.lastModifiedDate;
}

public RecordStatus getStatus() {
    return this.status;
}

public UUID getWorkspaceId() {
    return this.workspaceId;
}

public UUID getBranchId() {
    return this.branchId;
}

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

public void setCreatedBy(final String createdBy) {
    this.createdBy = createdBy;
}

public void setUpdatedBy(final String updatedBy) {
    this.updatedBy = updatedBy;
}

@JsonDeserialize(
    using = InstantJsonDeSerializer.class
)
```

```
    public void setCreatedDate(final Instant createdDate) {
        this.createdDate = createdDate;
    }

    @JsonDeserialize(
        using = InstantJsonDeSerializer.class
    )
    public void setLastModifiedDate(final Instant lastModifiedDate) {
        this.lastModifiedDate = lastModifiedDate;
    }

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

    public void setWorkspaceId(final UUID workspaceId) {
        this.workspaceId = workspaceId;
    }

    public void setBranchId(final UUID branchId) {
        this.branchId = branchId;
    }
}
```

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.lawcubator.falcon.infra.model;

import java.util.UUID;

public interface IModel {
    UUID getWorkspaceId();

    UUID getBranchId();
}
```

```
package com.lawcubator.falcon.lms.model.db.compliancepolicy;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.lawcubator.falcon.infra.annotations.DTOIgnore;
import com.lawcubator.falcon.infra.annotations.EntityOwners;
import com.lawcubator.falcon.infra.model.Model;
import com.lawcubator.falcon.infra.model.WorkspaceUserRole;
```

```java
import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.EntityListeners;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.OneToMany;
import lombok.Data;
import lombok.EqualsAndHashCode;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import java.util.List;

@EqualsAndHashCode(callSuper = true)
@Entity(name = "workspace_policy")
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
@EntityListeners(AuditingEntityListener.class)
@Data
@EntityOwners(roles = {WorkspaceUserRole.BRANCH_ADMIN,
WorkspaceUserRole.WORKSPACE_ADMIN, WorkspaceUserRole.EMPLOYER,
   WorkspaceUserRole.LAWCUBATOR_ADMIN})
public class WorkspacePolicy extends Model {
   @ManyToOne
   private Policy policy;

   @DTOIgnore
   @OneToMany(mappedBy = "workspacePolicy", cascade = CascadeType.REMOVE)
   private List<WorkspacePolicyAssign> workspacePolicyAssigns;

   @DTOIgnore
   @OneToMany(mappedBy = "workspacePolicy", cascade = CascadeType.REMOVE)
   private List<WorkspacePolicyAssignInfo> workspacePolicyAssignsInfo;

   @Column
   private boolean isAssignedToAllBranch;
}
```

```java
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.lawcubator.falcon.infra.model;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import com.lawcubator.falcon.infra.annotations.EntityOwners;
```

```java
import com.lawcubator.falcon.infra.core.InstantJsonDeSerializer;
import com.lawcubator.falcon.infra.core.InstantJsonSerializer;
import com.lawcubator.falcon.infra.enums.RecordStatus;
import jakarta.persistence.Column;
import jakarta.persistence.EnumType;
import jakarta.persistence.Enumerated;
import jakarta.persistence.Id;
import jakarta.persistence.MappedSuperclass;
import jakarta.persistence.PrePersist;
import java.time.Instant;
import java.util.UUID;
import org.springframework.data.annotation.CreatedBy;
import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.annotation.LastModifiedBy;
import org.springframework.data.annotation.LastModifiedDate;

@MappedSuperclass
@EntityOwners(
   roles = {WorkspaceUserRole.WORKSPACE_ADMIN}
)
public class Model implements IModel {
   @Id
   private UUID id;
   @CreatedBy
   private String createdBy;
   @LastModifiedBy
   private String updatedBy;
   @CreatedDate
   @JsonDeserialize(
       using = InstantJsonDeSerializer.class
   )
   @JsonSerialize(
       using = InstantJsonSerializer.class
   )
   private Instant createdDate;
   @LastModifiedDate
   @JsonDeserialize(
       using = InstantJsonDeSerializer.class
   )
   @JsonSerialize(
       using = InstantJsonSerializer.class
   )
   private Instant lastModifiedDate;
   @Enumerated(EnumType.STRING)
   private RecordStatus status;
   @Column
   private UUID workspaceId;
```

```
    @Column
    private UUID branchId;

    @PrePersist
    private void ensureId() {
        this.setId(UUID.randomUUID());
    }

    public Model() {
        this.status = RecordStatus.ACTIVE;
    }

    public Model(final UUID id, final String createdBy, final String updatedBy,
final Instant createdDate, final Instant lastModifiedDate, final RecordStatus
status, final UUID workspaceId, final UUID branchId) {
        this.status = RecordStatus.ACTIVE;
        this.id = id;
        this.createdBy = createdBy;
        this.updatedBy = updatedBy;
        this.createdDate = createdDate;
        this.lastModifiedDate = lastModifiedDate;
        this.status = status;
        this.workspaceId = workspaceId;
        this.branchId = branchId;
    }

    public UUID getId() {
        return this.id;
    }

    public String getCreatedBy() {
        return this.createdBy;
    }

    public String getUpdatedBy() {
        return this.updatedBy;
    }

    public Instant getCreatedDate() {
        return this.createdDate;
    }

    public Instant getLastModifiedDate() {
        return this.lastModifiedDate;
    }

    public RecordStatus getStatus() {
```

```
        return this.status;
    }

    public UUID getWorkspaceId() {
        return this.workspaceId;
    }

    public UUID getBranchId() {
        return this.branchId;
    }

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

    public void setCreatedBy(final String createdBy) {
        this.createdBy = createdBy;
    }

    public void setUpdatedBy(final String updatedBy) {
        this.updatedBy = updatedBy;
    }

    @JsonDeserialize(
        using = InstantJsonDeSerializer.class
    )
    public void setCreatedDate(final Instant createdDate) {
        this.createdDate = createdDate;
    }

    @JsonDeserialize(
        using = InstantJsonDeSerializer.class
    )
    public void setLastModifiedDate(final Instant lastModifiedDate) {
        this.lastModifiedDate = lastModifiedDate;
    }

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

    public void setWorkspaceId(final UUID workspaceId) {
        this.workspaceId = workspaceId;
    }

    public void setBranchId(final UUID branchId) {
        this.branchId = branchId;
```

```
    }
}
```

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.lawcubator.falcon.infra.model;

import java.util.UUID;

public interface IModel {
    UUID getWorkspaceId();

    UUID getBranchId();
}
```

```
package com.lawcubator.falcon.lms.model.db.compliancepolicy;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.lawcubator.falcon.infra.annotations.EntityOwners;
import com.lawcubator.falcon.infra.annotations.Validate;
import com.lawcubator.falcon.infra.model.Model;
import com.lawcubator.falcon.infra.model.WorkspaceUserRole;
import com.lawcubator.falcon.lms.constants.PolicyAttemptStatus;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.EntityListeners;
import jakarta.persistence.EnumType;
import jakarta.persistence.Enumerated;
import jakarta.persistence.ManyToOne;
import lombok.Data;
import lombok.EqualsAndHashCode;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import java.time.Instant;
import java.util.UUID;

@EqualsAndHashCode(callSuper = true)
@Entity(name = "workspace_policy_assign")
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
@EntityListeners(AuditingEntityListener.class)
@Data
@EntityOwners(roles = {WorkspaceUserRole.BRANCH_ADMIN,
WorkspaceUserRole.WORKSPACE_ADMIN, WorkspaceUserRole.EMPLOYEE,
```

```
    WorkspaceUserRole.EMPLOYER})
public class WorkspacePolicyAssign extends Model {
    @ManyToOne
    private WorkspacePolicy workspacePolicy;

    /*@Column(nullable = false)
    @Validate(nonNullable = true, nonEmpty = true, displayName = "Branch")
    private UUID branchId;*/

    @Validate(displayName = "Department")
    private UUID departmentId;

    @Column(nullable = false)
    @Validate(nonNullable = true, nonEmpty = true, displayName = "User")
    private UUID userId;

    @Column(nullable = false)
    @Validate(nonNullable = true, nonEmpty = true, displayName = "Attempt
Status")
    @Enumerated(EnumType.STRING)
    private PolicyAttemptStatus attemptStatus;

    private Instant approvedDate;
}
```

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.lawcubator.falcon.infra.model;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;
import com.lawcubator.falcon.infra.annotations.EntityOwners;
import com.lawcubator.falcon.infra.core.InstantJsonDeSerializer;
import com.lawcubator.falcon.infra.core.InstantJsonSerializer;
import com.lawcubator.falcon.infra.enums.RecordStatus;
import jakarta.persistence.Column;
import jakarta.persistence.EnumType;
import jakarta.persistence.Enumerated;
import jakarta.persistence.Id;
import jakarta.persistence.MappedSuperclass;
import jakarta.persistence.PrePersist;
import java.time.Instant;
import java.util.UUID;
```

```java
import org.springframework.data.annotation.CreatedBy;
import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.annotation.LastModifiedBy;
import org.springframework.data.annotation.LastModifiedDate;

@MappedSuperclass
@EntityOwners(
   roles = {WorkspaceUserRole.WORKSPACE_ADMIN}
)
public class Model implements IModel {
   @Id
   private UUID id;
   @CreatedBy
   private String createdBy;
   @LastModifiedBy
   private String updatedBy;
   @CreatedDate
   @JsonDeserialize(
       using = InstantJsonDeSerializer.class
   )
   @JsonSerialize(
       using = InstantJsonSerializer.class
   )
   private Instant createdDate;
   @LastModifiedDate
   @JsonDeserialize(
       using = InstantJsonDeSerializer.class
   )
   @JsonSerialize(
       using = InstantJsonSerializer.class
   )
   private Instant lastModifiedDate;
   @Enumerated(EnumType.STRING)
   private RecordStatus status;
   @Column
   private UUID workspaceId;
   @Column
   private UUID branchId;

   @PrePersist
   private void ensureId() {
       this.setId(UUID.randomUUID());
   }

   public Model() {
       this.status = RecordStatus.ACTIVE;
   }
```

```java
   public Model(final UUID id, final String createdBy, final String updatedBy,
final Instant createdDate, final Instant lastModifiedDate, final RecordStatus
status, final UUID workspaceId, final UUID branchId) {
       this.status = RecordStatus.ACTIVE;
       this.id = id;
       this.createdBy = createdBy;
       this.updatedBy = updatedBy;
       this.createdDate = createdDate;
       this.lastModifiedDate = lastModifiedDate;
       this.status = status;
       this.workspaceId = workspaceId;
       this.branchId = branchId;
   }

   public UUID getId() {
       return this.id;
   }

   public String getCreatedBy() {
       return this.createdBy;
   }

   public String getUpdatedBy() {
       return this.updatedBy;
   }

   public Instant getCreatedDate() {
       return this.createdDate;
   }

   public Instant getLastModifiedDate() {
       return this.lastModifiedDate;
   }

   public RecordStatus getStatus() {
       return this.status;
   }

   public UUID getWorkspaceId() {
       return this.workspaceId;
   }

   public UUID getBranchId() {
       return this.branchId;
   }
```

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

    public void setCreatedBy(final String createdBy) {
        this.createdBy = createdBy;
    }

    public void setUpdatedBy(final String updatedBy) {
        this.updatedBy = updatedBy;
    }

    @JsonDeserialize(
        using = InstantJsonDeSerializer.class
    )
    public void setCreatedDate(final Instant createdDate) {
        this.createdDate = createdDate;
    }

    @JsonDeserialize(
        using = InstantJsonDeSerializer.class
    )
    public void setLastModifiedDate(final Instant lastModifiedDate) {
        this.lastModifiedDate = lastModifiedDate;
    }

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

    public void setWorkspaceId(final UUID workspaceId) {
        this.workspaceId = workspaceId;
    }

    public void setBranchId(final UUID branchId) {
        this.branchId = branchId;
    }
}
```

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.lawcubator.falcon.infra.model;
```

```java
import java.util.UUID;

public interface IModel {
   UUID getWorkspaceId();

   UUID getBranchId();
}
```

```java
package com.lawcubator.falcon.lms.dtos.pojo.internal;


import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.util.List;
import java.util.UUID;


@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
public class BranchDepartmentInfo {
   private UUID branchId;
   private String branchName;
   private List<DepartmentResponse> departments;
}
```

```java
package com.lawcubator.falcon.lms.dtos.pojo.internal;


import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.util.UUID;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class DepartmentResponse {
   private UUID departmentId;
```

```
   private String departmentName;
}
```

```
package com.lawcubator.falcon.lms.dtos.request.lms;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.util.List;
import java.util.UUID;

@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
public class BranchInfoRequest {
   private UUID branchId;
   private boolean isAllDepartments;
   private List<UUID> departmentIds;
}
```

```
package com.lawcubator.falcon.lms.helper;

import com.lawcubator.falcon.infra.authentication.dto.UserTokenContext;
import com.lawcubator.falcon.infra.model.WorkspaceUserRole;
import com.lawcubator.falcon.lms.constants.AdminPermissionType;
import com.lawcubator.falcon.lms.constants.WorkspacePermission;
import com.lawcubator.falcon.lms.dtos.pojo.internal.Branch;
import com.lawcubator.falcon.lms.dtos.pojo.internal.BranchDepartmentInfo;
import com.lawcubator.falcon.lms.dtos.pojo.internal.DynamicUserRoutesDto;
import com.lawcubator.falcon.lms.dtos.pojo.internal.InternalAllUserResponse;
import
com.lawcubator.falcon.lms.dtos.pojo.internal.InternalBranchDepartmentInfo;
import com.lawcubator.falcon.lms.dtos.pojo.internal.InternalWorkspaceInfo;
import com.lawcubator.falcon.lms.dtos.pojo.user.UserInvite;
import com.lawcubator.falcon.lms.dtos.pojo.user.UserResponse;
import com.lawcubator.falcon.lms.dtos.pojo.user.WorkspaceUserResponseDto;
import com.lawcubator.falcon.lms.dtos.pojo.user.WorkspaceUserRoleResponseDto;
import com.lawcubator.falcon.lms.dtos.request.internal.InternalAllUserRequest;
import
com.lawcubator.falcon.lms.dtos.request.internal.InternalBranchDepartmentRequest
;
import com.lawcubator.falcon.lms.dtos.request.lms.BulkUserAssignRequest;
import com.lawcubator.falcon.lms.dtos.request.lms.WorkspaceUserRequest;
```

```java
import org.springframework.data.domain.Page;

import java.util.Collection;
import java.util.List;
import java.util.Optional;
import java.util.UUID;

public interface UserManagementApiHelperService {
    /**
     * Get user from a workspace.
     *
     * @param workspaceId workspaceId to search for the user.
     * @param userId      user to search in workspace.
     * @return WorkspaceUserResponseDto
     */
    WorkspaceUserResponseDto getUserData(UUID workspaceId, UUID userId);

    /**
     * Get user info by cognito user id.
     *
     * @param cognitoUserId cognito user id to search for.
     * @return UserResponse user info
     */
    UserResponse getUserByCognitoUserId(String cognitoUserId);

    /**
     * Get user info by user id.
     *
     * @param userId user id to search for.
     * @return UserResponse user info
     */
    UserResponse getActiveUserById(UUID userId);

    UserResponse getActiveOrInActiveUserByUserId(UUID userId);

    /**
     * Get list of users in workspace based on some filters.
     *
     * @param workspaceId          workspace to look for
     * @param workspaceUserRequest filter request to apply.
     * @return list of users present in workspace based on filter applied.
     */
    List<WorkspaceUserResponseDto> getWorkspaceUserData(UUID workspaceId,
                                                        WorkspaceUserRequest
workspaceUserRequest,
                                                        boolean
includeNonActiveRecords, boolean includeDeletedRecords);
```

```java
    /**
     * Get list of users in workspace.
     *
     * @param workspaceId workspace to look for.
     * @return list of users in workspace.
     */
    List<WorkspaceUserResponseDto> getWorkspaceUsers(UUID workspaceId, boolean
includeNonActiveRecords,
                                                     boolean
includeDeletedRecords);

    /**
     * Role information of user in workspace.
     *
     * @param workspaceId workspace this user present in.
     * @param userId      user to check the role for.
     * @return role details for this user.
     */
    WorkspaceUserRoleResponseDto getWorkspaceUserRoleDetails(UUID workspaceId,
UUID userId);

    /**
     * Get user present in workspace by list of email addresses.
     *
     * @param bulkUserAssignRequest request containing list of emails.
     * @param workspaceId           workspace to look into.
     * @return list of user info
     */
    List<WorkspaceUserResponseDto>
getUsersByEmailOrUserName(BulkUserAssignRequest bulkUserAssignRequest,
                                                            UUID workspaceId,
boolean includeNonActiveRecords);

    /**
     * Get branch and department information.
     *
     * @param request request containing branch id and department id.
     * @return Branch and department details.
     */
    InternalBranchDepartmentInfo
getBranchDepartmentInfo(InternalBranchDepartmentRequest request);

    /**
     * Get all users with all user information from user-management for report
apis.
     *
```

```
    * @param request    request containing filters
    * @param pageNumber page info
    * @param pageSize   page info
    * @return page of users.
    */
   Page<InternalAllUserResponse> getAllUsersForReport(InternalAllUserRequest
request,
                                                      int pageNumber, int
pageSize, boolean includeNonActiveRecords);


   boolean checkIfAdminHasPermission(UUID workspaceId, AdminPermissionType
adminPermissionType);

   boolean checkIfRoleHasPermission(UUID workspaceId,
                                    WorkspaceUserRole workspaceRole,
                                    WorkspacePermission workspacePermission);

   boolean isValidUserToken(UserTokenContext context);

   InternalWorkspaceInfo getWorkspaceInfoByWorkspaceId(UUID workspaceId);

   Optional<InternalWorkspaceInfo> getWorkspaceInfoByWorkspaceUrl(String
workspaceUrl);

   List<Branch> getAllBranchByWorkspaceId(UUID workspaceId);

   DynamicUserRoutesDto createDynamicUserRoute(DynamicUserRoutesDto
dynamicUserRoutesDto);

   String getWorkspaceUserInviteLink(UUID userId);

   List<UserResponse> getUsersPartOfWorkspace(Collection<UUID> userIds,
                                              UUID workspaceId, boolean
includeNonActiveRecords);

   List<UserInvite> getWorkspaceUserInviteLinksForInActiveUsers(UUID
workspaceId, List<UUID> userIds);

   List<DynamicUserRoutesDto>
bulkCreateDynamicUserRoutes(List<DynamicUserRoutesDto> dynamicUserRoutesDtos);

   List<InternalWorkspaceInfo> getWorkspaceInfo(List<UUID> workspaceIds);

   List<BranchDepartmentInfo> getBranchDepartments(UUID workspaceId);
}
```

```java
package com.lawcubator.falcon.lms.helper;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.core.type.TypeReference;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.lawcubator.falcon.infra.authentication.dto.UserTokenContext;
import com.lawcubator.falcon.infra.core.logging.Logger;
import com.lawcubator.falcon.infra.exceptions.ExceptionGenerator;
import com.lawcubator.falcon.infra.model.WorkspaceUserRole;
import com.lawcubator.falcon.lms.constants.AdminPermissionType;
import com.lawcubator.falcon.lms.constants.WorkspacePermission;
import com.lawcubator.falcon.lms.dtos.pojo.internal.Branch;
import com.lawcubator.falcon.lms.dtos.pojo.internal.BranchDepartmentInfo;
import com.lawcubator.falcon.lms.dtos.pojo.internal.DynamicUserRoutesDto;
import com.lawcubator.falcon.lms.dtos.pojo.internal.InternalAllUserResponse;
import
com.lawcubator.falcon.lms.dtos.pojo.internal.InternalBranchDepartmentInfo;
import
com.lawcubator.falcon.lms.dtos.pojo.internal.InternalWorkSpaceUserResponseDto;
import com.lawcubator.falcon.lms.dtos.pojo.internal.InternalWorkspaceInfo;
import com.lawcubator.falcon.lms.dtos.pojo.internal.WorkspaceInfoRequest;
import com.lawcubator.falcon.lms.dtos.pojo.user.UserInvite;
import com.lawcubator.falcon.lms.dtos.pojo.user.UserResponse;
import com.lawcubator.falcon.lms.dtos.pojo.user.WorkspaceUserResponseDto;
import com.lawcubator.falcon.lms.dtos.pojo.user.WorkspaceUserRoleResponseDto;
import
com.lawcubator.falcon.lms.dtos.request.internal.InternalAdminPermissionRequest;
import com.lawcubator.falcon.lms.dtos.request.internal.InternalAllUserRequest;
import
com.lawcubator.falcon.lms.dtos.request.internal.InternalBranchDepartmentRequest
;
import
com.lawcubator.falcon.lms.dtos.request.internal.InternalPermissionCheckRequest;
import com.lawcubator.falcon.lms.dtos.request.lms.BulkUserAssignRequest;
import com.lawcubator.falcon.lms.dtos.request.lms.WorkspaceUserRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.domain.Page;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.util.UriComponentsBuilder;
```

```java
import java.net.URI;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.List;
import java.util.Objects;
import java.util.Optional;
import java.util.UUID;

@Service
public class UmsApiHelperServiceImpl implements UserManagementApiHelperService
{

   private final ObjectMapper objectMapper;
   private final ExceptionGenerator exceptionGenerator;

   @Value("${usermanagement.base.url}")
   private String userManagementBaseApiUrl;

   @Value("${usermanagement.base.by-cognito-user-id-url}")
   private String userManagementUserByCognitoUserId;

   @Value("${usermanagement.base.branch-info-url}")
   private String branchInfoUrl;

   @Value("${usermanagement.base.valid-role-url}")
   private String userRoleInfoUrl;

   @Value("${usermanagement.base.user-by-user-id}")
   private String userByUserIdUrl;

   @Value("${usermanagement.base.get-all-branch}")
   private String allBranchUrl;

   @Value("${usermanagement.base.users-for-report}")
   private String usersForReportUrl;

   @Value("${usermanagement.base.admin-permission-url}")
   private String adminPermissionUrl;

   @Value("${usermanagement.base.workspace-permission-check}")
   private String workspacePermissionCheck;

   @Value("${usermanagement.base.validate-token}")
   private String tokenValidityUrl;
```

```
    @Value("${usermanagement.base.user-dynamic-routes}")
    private String dynamicUserRoutesUrl;

    @Value("${usermanagement.base.workspace-info-list}")
    private String workspaceInfoListUrl;

    @Value("${usermanagement.base.workspace-url}")
    private String workspaceUrl;

    @Value("${usermanagement.base.user-invite-link}")
    private String inviteLinkForUserUrl;

    @Value("${usermanagement.base.users-in-workspace}")
    private String usersInWorkspaceUrl;

    @Value("${usermanagement.base.inactive-users-invites}")
    private String bulkInActiveUserInviteLink;

    @Value("${usermanagement.base.bulk-user-dynamic-routes}")
    private String bulkCreateUserDynamicRoutes;

    private final RestTemplate restTemplate;

    @Autowired
    public UmsApiHelperServiceImpl(
        ObjectMapper objectMapper,
        @Qualifier("falconRestTemplate") RestTemplate restTemplate,
        ExceptionGenerator exceptionGenerator) {
        this.objectMapper = objectMapper;
        this.exceptionGenerator = exceptionGenerator;
        this.restTemplate = restTemplate;
    }

    @Override
    public WorkspaceUserResponseDto getUserData(UUID workspaceId, UUID userId) {

        return restTemplate.getForObject(this.userManagementBaseApiUrl +
workspaceId + "/user/" + userId,
            WorkspaceUserResponseDto.class);
    }

    @Override
    public UserResponse getUserByCognitoUserId(String cognitoUserId) {

        return restTemplate.getForObject(this.userManagementUserByCognitoUserId
+ cognitoUserId,
            UserResponse.class);
```

```
    }

    @Override
    public UserResponse getActiveUserById(UUID userId) {
        try {

            return restTemplate.getForObject(this.userByUserIdUrl + userId,
                UserResponse.class);
        } catch (Exception e) {
            return null;
        }
    }


    public UserResponse getActiveOrInActiveUserByUserId(UUID userId) {
        try {

            return restTemplate.getForObject(this.userByUserIdUrl + userId
                                                    +
"?includeNonActiveRecords={includeNonActiveRecords}",
                UserResponse.class, true);
        } catch (Exception e) {
            return null;
        }
    }

    @Override
    public List<WorkspaceUserResponseDto> getWorkspaceUserData(UUID workspaceId,

WorkspaceUserRequest workspaceUserRequest,
                                                               boolean
includeNonActiveRecords,
                                                               boolean
includeDeletedRecords) {

        HttpEntity<WorkspaceUserRequest> entity = new
HttpEntity<WorkspaceUserRequest>(workspaceUserRequest);

        ResponseEntity<String> response = restTemplate
            .exchange(this.userManagementBaseApiUrl + workspaceId
                      +
"/workspace-user-details?includeNonActiveRecords={includeNonActiveRecords"
                      + "}&includeDeletedRecords={includeDeletedRecords}",
                HttpMethod.POST, entity,
                String.class, includeNonActiveRecords, includeDeletedRecords);

        String res = response.getBody();
```

```
        if (Objects.nonNull(res)) {
            InternalWorkSpaceUserResponseDto internalWorkSpaceUserResponseDto =
null;
            try {
                internalWorkSpaceUserResponseDto =
                    objectMapper.readValue(res, new TypeReference<>() {
                    });
            } catch (JsonProcessingException e) {
                e.getMessage();
            }
            return internalWorkSpaceUserResponseDto.getWorkspaceUserDtoList();
        }
        throw exceptionGenerator.generateNotFoundException("User Details");
    }

    @Override
    public List<WorkspaceUserResponseDto> getWorkspaceUsers(UUID workspaceId,
boolean includeNonActiveRecords,
                                                             boolean
includeDeletedRecords) {

        ResponseEntity<String> response = restTemplate
            .getForEntity(this.userManagementBaseApiUrl + workspaceId
                          +
"/workspace-users?includeNonActiveRecords={includeNonActiveRecords}&includeDele
tedRecords"
                          + "={includeDeletedRecords}",
                String.class, includeNonActiveRecords, includeDeletedRecords);
        String res = response.getBody();
        if (Objects.nonNull(res)) {
            InternalWorkSpaceUserResponseDto internalWorkSpaceUserResponseDto =
null;
            try {
                internalWorkSpaceUserResponseDto =
                    objectMapper.readValue(res, new TypeReference<>() {
                    });
            } catch (JsonProcessingException e) {
                e.getMessage();
            }
            return internalWorkSpaceUserResponseDto.getWorkspaceUserDtoList();
        }
        throw exceptionGenerator.generateNotFoundException("User Details");
    }

    @Override
    public WorkspaceUserRoleResponseDto getWorkspaceUserRoleDetails(UUID
workspaceId, UUID userId) {
```

```
        ///{workspaceId}/user/{userId}/user-role-details

        ResponseEntity<String> response = restTemplate
            .getForEntity(this.userManagementBaseApiUrl + workspaceId + "/user/"
+ userId + "/user-role-details",
                String.class);

        String res = response.getBody();
        if (Objects.isNull(res)) {
            throw exceptionGenerator.generateNotFoundException("USER Details");
        }
        WorkspaceUserRoleResponseDto workspaceUserRoleResponseDto = null;
        try {
            workspaceUserRoleResponseDto =
                objectMapper.readValue(res, new TypeReference<>() {
                });
        } catch (JsonProcessingException e) {
            e.getMessage();
        }
        return workspaceUserRoleResponseDto;
    }

    @Override
    public List<WorkspaceUserResponseDto>
getUsersByEmailOrUserName(BulkUserAssignRequest bulkUserAssignRequest,
                                                                    UUID
workspaceId, boolean includeNonActiveRecords) {

        HttpEntity<BulkUserAssignRequest> entity = new
HttpEntity<BulkUserAssignRequest>(bulkUserAssignRequest);

        ResponseEntity<String> response = restTemplate
            .exchange(this.userManagementBaseApiUrl + workspaceId
                    +
"/get-users?includeNonActiveRecords={includeNonActiveRecords}",
HttpMethod.POST, entity,
                String.class, includeNonActiveRecords);

        String res = response.getBody();
        List<WorkspaceUserResponseDto> workspaceUserResponseDtoList = new
ArrayList<>();
        if (Objects.nonNull(res)) {
            try {
                workspaceUserResponseDtoList =
                    objectMapper.readValue(res, new
TypeReference<List<WorkspaceUserResponseDto>>() {
                    });
```

```
            } catch (JsonProcessingException e) {
                e.getMessage();
            }

        }
        return workspaceUserResponseDtoList;
    }

    @Override
    public InternalBranchDepartmentInfo
getBranchDepartmentInfo(InternalBranchDepartmentRequest request) {

        HttpEntity<InternalBranchDepartmentRequest> entity = new
HttpEntity<>(request);

        ResponseEntity<String> response = restTemplate
            .exchange(this.branchInfoUrl, HttpMethod.POST, entity,
                String.class);

        String res = response.getBody();
        InternalBranchDepartmentInfo internalBranchDepartmentInfo = null;
        if (Objects.nonNull(res)) {
            try {
                internalBranchDepartmentInfo = objectMapper.readValue(res, new
TypeReference<>() {
                });
            } catch (JsonProcessingException e) {
                e.getMessage();
            }
        }
        return internalBranchDepartmentInfo;
    }

    @Override
    public Page<InternalAllUserResponse>
getAllUsersForReport(InternalAllUserRequest request,
                                                                int pageNumber,
int pageSize,
                                                                boolean
includeNonActiveRecords) {

        HttpEntity<InternalAllUserRequest> entity = new HttpEntity<>(request);

        URI uri = UriComponentsBuilder
            .fromUriString(this.usersForReportUrl)
            .queryParam("pageNo", pageNumber)
            .queryParam("pageSize", pageSize)
```

```
            .queryParam("includeNonActiveRecords", includeNonActiveRecords)
            .build()
            .toUri();

        ResponseEntity<String> response = restTemplate.exchange(uri,
HttpMethod.POST, entity, String.class);

        String res = response.getBody();
        Page<InternalAllUserResponse> userResponsePage = null;
        if (Objects.nonNull(res)) {
            try {
                userResponsePage =
                    objectMapper.readValue(res, new
TypeReference<RestResponsePage<InternalAllUserResponse>>() {
                    });
            } catch (JsonProcessingException e) {
                e.getMessage();
            }

        }
        return userResponsePage;
    }

    @Override
    public boolean checkIfAdminHasPermission(UUID workspaceId,
AdminPermissionType adminPermissionType) {
        InternalAdminPermissionRequest request = new
InternalAdminPermissionRequest();
        request.setPermissionType(adminPermissionType);
        request.setWorkspaceId(workspaceId);

        HttpEntity<InternalAdminPermissionRequest> entity = new
HttpEntity<>(request);

        URI uri = UriComponentsBuilder
            .fromUriString(this.adminPermissionUrl)
            .build()
            .toUri();

        ResponseEntity<String> response = restTemplate.exchange(uri,
HttpMethod.POST, entity, String.class);

        String res = response.getBody();
        boolean isValidPermission = false;
        if (Objects.nonNull(res)) {
            try {
```

```
                isValidPermission = objectMapper.readValue(res, new
TypeReference<>() {
                });
            } catch (JsonProcessingException e) {
                e.getMessage();
            }

        }
        return isValidPermission;
    }

    @Override
    public boolean checkIfRoleHasPermission(UUID workspaceId,
                                            WorkspaceUserRole workspaceRole,
                                            WorkspacePermission
workspacePermission) {
        InternalPermissionCheckRequest request = new
InternalPermissionCheckRequest();
        request.setWorkspaceId(workspaceId);
        request.setWorkspacePermission(workspacePermission);
        request.setWorkspaceRole(workspaceRole);


        HttpEntity<InternalPermissionCheckRequest> entity = new
HttpEntity<>(request);
        boolean isValidPermission = false;

        try {
            ResponseEntity<Boolean> response = restTemplate
                .exchange(this.workspacePermissionCheck, HttpMethod.POST,
entity,
                    Boolean.class);
            if (response.getBody() != null) {
                isValidPermission = response.getBody();
            }
        } catch (Exception ignored) {
            isValidPermission = false;
        }

        return isValidPermission;
    }

    @Override
    public boolean isValidUserToken(UserTokenContext context) {

        HttpEntity<UserTokenContext> entity = new HttpEntity<>(context);
        boolean isValidToken = false;
```

```
        try {
            ResponseEntity<Boolean> response = restTemplate
                .exchange(this.tokenValidityUrl, HttpMethod.POST, entity,
                    Boolean.class);
            if (response.getBody() != null) {
                isValidToken = response.getBody();
            }
        } catch (Exception ignored) {
            isValidToken = false;
        }

        return isValidToken;
    }

    @Override
    public InternalWorkspaceInfo getWorkspaceInfoByWorkspaceId(UUID workspaceId)
{

        try {
            return restTemplate.getForObject(this.userManagementBaseApiUrl +
workspaceId,
                InternalWorkspaceInfo.class);
        } catch (Exception e) {
            return null;
        }
    }

    @Override
    public Optional<InternalWorkspaceInfo> getWorkspaceInfoByWorkspaceUrl(String
workspaceUrl) {
        return Optional.ofNullable(restTemplate.getForObject(this.workspaceUrl +
workspaceUrl,
            InternalWorkspaceInfo.class));
    }

    @Override
    public List<Branch> getAllBranchByWorkspaceId(UUID workspaceId) {

        ResponseEntity<String> response = restTemplate
            .getForEntity(this.allBranchUrl + workspaceId, String.class);
        String res = response.getBody();
        if (Objects.nonNull(res)) {
            List<Branch> allBranches = Collections.emptyList();
            try {
                allBranches = objectMapper.readValue(res, new TypeReference<>()
{
```

```
                });
            } catch (JsonProcessingException e) {
                e.getMessage();
            }
            return allBranches;
        }
        return Collections.emptyList();
    }

    @Override
    public DynamicUserRoutesDto createDynamicUserRoute(DynamicUserRoutesDto
dynamicUserRoutesDto) {
        URI uri = UriComponentsBuilder
            .fromUriString(this.dynamicUserRoutesUrl)
            .build()
            .toUri();

        HttpEntity<DynamicUserRoutesDto> entity = new
HttpEntity<>(dynamicUserRoutesDto);
        ResponseEntity<DynamicUserRoutesDto> response =
            restTemplate.exchange(uri, HttpMethod.POST, entity,
DynamicUserRoutesDto.class);
        return response.getBody();
    }

    @Override
    public String getWorkspaceUserInviteLink(UUID userId) {
        return restTemplate.getForObject(this.inviteLinkForUserUrl + userId,
String.class);
    }

    @Override
    public List<UserResponse> getUsersPartOfWorkspace(Collection<UUID> userIds,
UUID workspaceId,
                                                      boolean
includeNonActiveRecords) {
        BulkUserAssignRequest bulkUserAssignRequest = new
BulkUserAssignRequest();
        bulkUserAssignRequest.setUserIds(new ArrayList<>(userIds));
        bulkUserAssignRequest.setWorkspaceId(workspaceId);
        bulkUserAssignRequest.setUserNames(Collections.emptyList());
        bulkUserAssignRequest.setEmailList(Collections.emptyList());
        bulkUserAssignRequest.setCourseIdList("");
        HttpEntity<BulkUserAssignRequest> entity = new
HttpEntity<>(bulkUserAssignRequest);

        URI uri = UriComponentsBuilder
```

```
            .fromUriString(this.usersInWorkspaceUrl)
            .queryParam("includeNonActiveRecords", includeNonActiveRecords)
            .build()
            .toUri();

        ResponseEntity<String> response = restTemplate.exchange(uri,
HttpMethod.POST, entity, String.class);

        String res = response.getBody();
        List<UserResponse> userResponsePage = null;
        if (Objects.nonNull(res)) {
            try {
                userResponsePage =
                    objectMapper.readValue(res, new
TypeReference<List<UserResponse>>() {
                    });
            } catch (JsonProcessingException e) {
                Logger.logError(this.getClass(), "getUsersPartOfWorkspace", e);
            }

        }
        return userResponsePage;
    }

    @Override
    public List<UserInvite> getWorkspaceUserInviteLinksForInActiveUsers(UUID
workspaceId, List<UUID> userIds) {
        BulkUserAssignRequest bulkUserAssignRequest = new
BulkUserAssignRequest();
        bulkUserAssignRequest.setUserIds(new ArrayList<>(userIds));
        bulkUserAssignRequest.setWorkspaceId(workspaceId);
        bulkUserAssignRequest.setUserNames(Collections.emptyList());
        bulkUserAssignRequest.setEmailList(Collections.emptyList());
        bulkUserAssignRequest.setCourseIdList("");
        HttpEntity<BulkUserAssignRequest> entity = new
HttpEntity<>(bulkUserAssignRequest);

        URI uri = UriComponentsBuilder
            .fromUriString(this.bulkInActiveUserInviteLink)
            .build()
            .toUri();

        ResponseEntity<String> response = restTemplate.exchange(uri,
HttpMethod.POST, entity, String.class);

        String res = response.getBody();
        List<UserInvite> userInvites = null;
```

```
        if (Objects.nonNull(res)) {
            try {
                userInvites = objectMapper.readValue(res, new
TypeReference<List<UserInvite>>() {
                });
            } catch (JsonProcessingException e) {
                Logger.logError(this.getClass(),
"getWorkspaceUserInviteLinksForInActiveUsers", e);
            }
        }
        return userInvites;
    }

    @Override
    public List<DynamicUserRoutesDto>
bulkCreateDynamicUserRoutes(List<DynamicUserRoutesDto> dynamicUserRoutesDtos) {

        HttpEntity<List<DynamicUserRoutesDto>> entity = new
HttpEntity<>(dynamicUserRoutesDtos);

        URI uri = UriComponentsBuilder
            .fromUriString(this.bulkCreateUserDynamicRoutes)
            .build()
            .toUri();

        ResponseEntity<String> response = restTemplate.exchange(uri,
HttpMethod.POST, entity, String.class);

        String res = response.getBody();
        List<DynamicUserRoutesDto> userInvites = null;
        if (Objects.nonNull(res)) {
            try {
                userInvites = objectMapper.readValue(res, new
TypeReference<List<DynamicUserRoutesDto>>() {
                });
            } catch (JsonProcessingException e) {
                Logger.logError(this.getClass(), "bulkCreateDynamicUserRoutes",
e);
            }
        }
        return userInvites;
    }

    @Override
    public List<InternalWorkspaceInfo> getWorkspaceInfo(List<UUID> workspaceIds)
{
        WorkspaceInfoRequest workspaceInfoRequest = new WorkspaceInfoRequest();
```

```
        workspaceInfoRequest.setWorkspaceIdList(workspaceIds);
        HttpEntity<WorkspaceInfoRequest> entity = new
HttpEntity<>(workspaceInfoRequest);
        URI uri = UriComponentsBuilder
            .fromUriString(this.workspaceInfoListUrl)
            .build()
            .toUri();

        List<InternalWorkspaceInfo> workspaceInfoList = null;
        try {
            ResponseEntity<String> response = restTemplate.exchange(uri,
HttpMethod.POST, entity, String.class);
            String res = response.getBody();
            workspaceInfoList = objectMapper.readValue(res, new
TypeReference<>() {
            });
        } catch (JsonProcessingException e) {
            Logger.logError(this.getClass(), "getWorkspaceInfo", e);
        }
        return workspaceInfoList;
    }

    @Override
    public List<BranchDepartmentInfo> getBranchDepartments(UUID workspaceId) {

        ResponseEntity<String> response = restTemplate
            .getForEntity(this.workspaceUrl + workspaceId +
"/all/branches/departments", String.class);

        String res = response.getBody();
        List<BranchDepartmentInfo> branchDepartmentInfo = new ArrayList<>();
        if (Objects.nonNull(res)) {
            try {
                branchDepartmentInfo = objectMapper.readValue(res, new
TypeReference<List<BranchDepartmentInfo>>() {

                });
            } catch (JsonProcessingException e) {
                Logger.logError(this.getClass(), "getBranchDepartments", e);
            }
        }
        return branchDepartmentInfo;
    }
}
```

```
package com.lawcubator.falcon.lms.repository.compliancepolicy;
```

```
import com.lawcubator.falcon.infra.annotations.WorkspaceSecureDisabled;
import com.lawcubator.falcon.infra.enums.RecordStatus;
import com.lawcubator.falcon.infra.core.repo.sql.FalconSqlRepository;
import com.lawcubator.falcon.lms.constants.ContentStatus;
import com.lawcubator.falcon.lms.model.db.compliancepolicy.Policy;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;
import java.util.Optional;
import java.util.UUID;

@Repository
@Transactional
public interface PolicyRepository extends FalconSqlRepository<Policy, UUID> {

   @WorkspaceSecureDisabled(any = true)
   Page<Policy> findByIsLawcubatorOwnedAndStatusAndPolicyStatus(boolean
isLawcubatorOwned,
                                                                RecordStatus
status,
                                                                ContentStatus
policyStatus,
                                                                Pageable
pageable);

   @WorkspaceSecureDisabled(any = true)
   Page<Policy>
findByIsLawcubatorOwnedAndStatusAndPolicyStatusAndPolicyCompliancesComplianceId
(
       boolean isLawcubatorOwned,
       RecordStatus status,
       ContentStatus policyStatus,
       UUID complianceId,
       Pageable pageable);

   @WorkspaceSecureDisabled(any = true)
   Page<Policy>
findByIsLawcubatorOwnedAndPolicyNameContainingIgnoreCaseOrPolicyDescriptionCont
ainingIgnoreCaseAndStatus(
       boolean isLawcubatorOwned, String policyName, String policyDescription,
RecordStatus recordStatus,
```

```
        Pageable pageable);

    @WorkspaceSecureDisabled(any = true)
    Page<Policy>
findByIsLawcubatorOwnedAndPolicyNameContainingIgnoreCaseAndStatus(
        boolean isLawcubatorOwned, String policyName, RecordStatus recordStatus,
Pageable pageable);

    @WorkspaceSecureDisabled(lawcubatorAdmin = true)
    Optional<Policy> findByIdAndStatus(UUID policyId, RecordStatus status);

    @Query(value =
        "select p.id from policy p where p.status=:status and p.id in (select
wp1.policy from workspace_policy wp1 "
        + "where wp1.id in "
        + "(select psca.workspacePolicy from workspace_policy_assign psca where
psca.workspacePolicy in "
        + "(select wp from workspace_policy wp where
wp.workspaceId=:workspaceId) and psca.userId=:userId))")
    List<UUID>
    getPolicyListByUser(@Param("workspaceId")
                        UUID workspaceId, @Param("userId") UUID userId,
@Param("status") RecordStatus status);
}
```

Class PolicyService :-

```
@Override
public Optional<Policy> getById(UUID policyId) {
    return policyRepository.findByIdAndStatus(policyId, RecordStatus.ACTIVE);
}
```

```
package com.lawcubator.falcon.lms.repository.compliancepolicy;

import com.lawcubator.falcon.infra.annotations.WorkspaceSecureDisabled;
import com.lawcubator.falcon.infra.enums.RecordStatus;
import com.lawcubator.falcon.infra.core.repo.sql.FalconSqlRepository;
import com.lawcubator.falcon.lms.constants.ContentStatus;
import com.lawcubator.falcon.lms.model.db.compliancepolicy.Policy;
import com.lawcubator.falcon.lms.model.db.compliancepolicy.WorkspacePolicy;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;
```

```
import java.time.Instant;
import java.util.Collection;
import java.util.List;
import java.util.Optional;
import java.util.UUID;

@Repository
@Transactional
public interface WorkspacePolicyRepository extends
FalconSqlRepository<WorkspacePolicy, UUID> {
   Page<WorkspacePolicy> findByWorkspaceIdAndStatus(UUID workspaceId,
                                                    RecordStatus status,
                                                    Pageable pageable);

   Optional<WorkspacePolicy> findByWorkspaceIdAndPolicyIdAndStatus(UUID
workspaceId,
                                                                   UUID
policyId, RecordStatus status);

   Optional<WorkspacePolicy>
findByWorkspaceIdAndPolicyIdAndPolicyPolicyStatusAndStatus(UUID workspaceId,

UUID policyId,

ContentStatus contentStatus,

RecordStatus status);

   long countByWorkspaceIdAndCreatedDateBetweenAndStatus(UUID workspaceId,
                                                         Instant startDate,
                                                         Instant endDate,
                                                         RecordStatus status);

   long countByWorkspaceIdAndStatus(UUID workspaceId, RecordStatus status);

   @Query(value =
       "select count(wp) from workspace_policy wp where
wp.workspaceId=:workspaceId and wp.branchId=:branchId and wp"
       + ".createdDate between :startDate and :endDate "
       + " and wp.status=:status")
   long
getCountByWorkspaceIdAndBranchIdAndCreatedDateBetween(@Param("workspaceId")
UUID workspaceId,

@Param("branchId") UUID branchId,

@Param("startDate") Instant startDate,
```

```
                                                                    @Param("endDate")
Instant endDate,
                                                                    @Param("status")
RecordStatus status);

   @Query(value =
       "select count(wp) from workspace_policy wp where
wp.workspaceId=:workspaceId "
       + " and wp.branchId=:branchId and wp.status=:status")
   long getCountByWorkspaceIdAndBranchId(@Param("workspaceId") UUID
workspaceId,
                                         @Param("branchId") UUID branchId,
                                         @Param("status") RecordStatus status);

   Page<WorkspacePolicy>
findByWorkspaceIdAndStatusAndPolicyPolicyCompliancesComplianceId(
       UUID workspaceId,
       RecordStatus status,
       UUID complianceId,
       Pageable pageable);

   Page<WorkspacePolicy>

findByWorkspaceIdAndPolicyPolicyCompliancesComplianceIdAndPolicyPolicyNameIgnor
eCaseContainingAndStatusOrWorkspaceIdAndPolicyPolicyCompliancesComplianceIdAndP
olicyPolicyDescriptionIgnoreCaseContainingAndStatus(
       UUID workspaceId,
       UUID complianceId,
       String searchTextName,
       RecordStatus recordStatus,
       UUID workspaceId1,
       UUID complianceId1,
       String searchTextDesc,
       RecordStatus recordStatus1,
       Pageable pageable);

   Page<WorkspacePolicy>

findByWorkspaceIdAndStatusAndPolicyPolicyNameIgnoreCaseContainingOrWorkspaceIdA
ndStatusAndPolicyPolicyDescriptionIgnoreCaseContaining(
       UUID workspaceId,
       RecordStatus status,
       String searchTextName,
       UUID workspaceId2,
       RecordStatus status2,
       String searchTextDesc,
       Pageable pageable);
```

```
   Page<WorkspacePolicy> findByWorkspaceIdAndBranchIdInAndStatus(
       UUID workspaceId,
       Collection<UUID> branchIds,
       RecordStatus status,
       Pageable pageable);

   Page<WorkspacePolicy>
findByWorkspaceIdAndBranchIdInAndPolicyPolicyNameIgnoreCaseContainingAndStatusO
rWorkspaceIdAndBranchIdInAndPolicyPolicyDescriptionIgnoreCaseContainingAndStatu
s(
       UUID workspaceId,
       Collection<UUID> branchIds,
       String searchTextName,
       RecordStatus status,
       UUID workspaceId1,
       Collection<UUID> branchIds1,
       String searchTextDesc,
       RecordStatus status1,
       Pageable pageable);

   Page<WorkspacePolicy>
findByWorkspaceIdAndBranchIdInAndPolicyPolicyCompliancesComplianceIdAndStatus(
       UUID workspaceId,
       Collection<UUID> branchIds,
       UUID complianceId,
       RecordStatus status,
       Pageable pageable);

   Page<WorkspacePolicy>
findByWorkspaceIdAndBranchIdInAndPolicyPolicyCompliancesComplianceIdAndPolicyPo
licyNameIgnoreCaseContainingAndStatusOrWorkspaceIdAndBranchIdInAndPolicyPolicyC
ompliancesComplianceIdAndPolicyPolicyDescriptionIgnoreCaseContainingAndStatus(
       UUID workspaceId,
       Collection<UUID> branchIds,
       UUID complianceId,
       String searchTextName,
       RecordStatus recordStatus,
       UUID workspaceId1,
       Collection<UUID> branchIds1,
       UUID complianceId1,
       String searchTextDesc,
       RecordStatus status,
       Pageable pageable);

   @Query(value = "select wp.policy.id from workspace_policy wp where
wp.workspaceId=:workspaceId and wp"
```

```
                + ".status=:status")
   List<UUID> getListOfPolicyByWorkspaceId(@Param("workspaceId") UUID
workspaceId,
                                           @Param("status") RecordStatus
status);

   @Query(value =
       "select distinct wspa.workspacePolicy.policy.id from
workspace_policy_assign wspa where wspa.status=:status"
       + " and wspa.branchId=:branchId and wspa.workspacePolicy in (select wsp
from workspace_policy wsp "
       + "where wsp.workspaceId=:workspaceId and wsp.status=:status)")
   List<UUID> getListOfPolicyByWorkspaceIdAndBranchId(@Param("workspaceId")
UUID workspaceId,
                                                      @Param("branchId") UUID
branchId,
                                                      @Param("status")
RecordStatus status);

   List<WorkspacePolicy> findByPolicyIdAndStatus(UUID policyId, RecordStatus
status);

   long countByWorkspaceIdAndPolicyIsLawcubatorOwnedAndStatus(UUID workspaceId,
boolean isLawcubatorOwned,
                                                              RecordStatus
status);

   List<WorkspacePolicy> findByPolicyInAndStatus(List<Policy> policyList,
RecordStatus status);

   List<WorkspacePolicy> findByPolicyIn(List<Policy> policyList);


   @WorkspaceSecureDisabled(lawcubatorAdmin = true)
   Page<WorkspacePolicy> findByWorkspaceId(UUID workspaceId, Pageable
pageable);

   @WorkspaceSecureDisabled(lawcubatorAdmin = true)
   List<WorkspacePolicy> findAllByIdInAndStatus(List<UUID>
workspacePolicyIdList, RecordStatus status);

   List<WorkspacePolicy>
findByWorkspaceIdAndIsAssignedToAllBranchAndStatus(UUID workspaceId,

boolean assignedToAllBranch,

RecordStatus status);
```

```
   @WorkspaceSecureDisabled(lawcubatorAdmin = true)
   Optional<WorkspacePolicy> findByIdAndStatus(UUID workspacePolicyId,
RecordStatus status);
}
```

```
package com.lawcubator.falcon.lms.repository.compliancepolicy;

import com.lawcubator.falcon.infra.annotations.WorkspaceSecureDisabled;
import com.lawcubator.falcon.infra.core.repo.sql.FalconSqlRepository;
import com.lawcubator.falcon.infra.enums.RecordStatus;
import com.lawcubator.falcon.lms.model.db.compliancepolicy.WorkspacePolicy;
import
com.lawcubator.falcon.lms.model.db.compliancepolicy.WorkspacePolicyAssignInfo;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;
import java.util.UUID;

@Repository
@Transactional
public interface WorkspacePolicyAssignInfoRepository extends
FalconSqlRepository<WorkspacePolicyAssignInfo, UUID> {


   @WorkspaceSecureDisabled(any = true)
   List<WorkspacePolicyAssignInfo>
findByWorkspacePolicyWorkspaceIdAndBranchIdAndStatus(UUID workspaceId,

UUID branchId,

RecordStatus status);

   @WorkspaceSecureDisabled(lawcubatorAdmin = true)
   List<WorkspacePolicyAssignInfo>
findByWorkspacePolicyAndStatus(WorkspacePolicy workspacePolicy,
                                                                  RecordStatus
status);

   @WorkspaceSecureDisabled(any = true)
   List<WorkspacePolicyAssignInfo>
findByWorkspacePolicyWorkspaceIdAndBranchIdAndDepartmentIdAndStatus(
       UUID workspaceId,
```

```
        UUID branchId,
        UUID departmentId,
        RecordStatus status);

    List<WorkspacePolicyAssignInfo>
findByWorkspacePolicyAndWorkspaceIdAndBranchIdAndStatus(
        WorkspacePolicy workspacePolicy, UUID workspaceId, UUID branchId,
RecordStatus status);

    List<WorkspacePolicyAssignInfo>
findByWorkspacePolicyIdInAndWorkspaceIdAndBranchIdAndDepartmentIdAndStatus(
        List<UUID> workspacePolicyIdList, UUID workspaceId, UUID branchId, UUID
departmentId, RecordStatus active);

    @Modifying
    @Query(value =
        "update workspace_policy_assign_info wpai set wpai.status=:newStatus
where wpai.workspacePolicy.id in "
        + ":workspacePolicyIdList and wpai.workspaceId=:workspaceId and
wpai.branchId=:branchId and "
        + "wpai.departmentId is null and wpai.status=:currentStatus")
    void updatePolicyAssignInfoForWorkspaceIdAndBranchIdAndStatus(List<UUID>
workspacePolicyIdList, UUID workspaceId,
                                                                  UUID branchId,
RecordStatus newStatus,
                                                                  RecordStatus
currentStatus);

    List<WorkspacePolicyAssignInfo>
findByWorkspaceIdAndIsForAllBranchesAndStatus(UUID workspaceId,

boolean isForAllBranch,

RecordStatus active);
}
```

WorkspacePolicyService :-

```
@Override
public void updateAutoAssignCapability(WorkspaceContext workspaceContext, UUID
policyId,
                                       AutoAssignBranchDepartmentRequest
request) {
    Optional<Policy> optionalPolicy = policyService.getById(policyId);
    if (optionalPolicy.isEmpty()) {
        throw getExceptionGenerator().generateNotFoundException("Policy");
```

```
    }

workspacePolicyAssignInfoService.supersedeExistingAndCreateNewAutoAssign(worksp
aceContext, policyId, request);
}
```

```
@Override
public Optional<WorkspacePolicy> getByWorkspaceIdAndPolicyId(UUID workspaceId,
UUID policyId) {
    return
workspacePolicyRepository.findByWorkspaceIdAndPolicyIdAndStatus(workspaceId,
policyId,
        RecordStatus.ACTIVE);
}
```

WorkspacePolicyAssignInfoService :-

```
@Override
public void supersedeExistingAndCreateNewAutoAssign(WorkspaceContext
workspaceContext, UUID policyId,

AutoAssignBranchDepartmentRequest request) {
    Optional<WorkspacePolicy> optionalWorkspacePolicy =

workspacePolicyService.getByWorkspaceIdAndPolicyId(workspaceContext.getWorkspac
eId(), policyId);
    if (optionalWorkspacePolicy.isEmpty()) {
        throw getExceptionGenerator().generateBadRequestException("Workspace
Policy");
    }

    List<BranchDepartmentInfo> branchDepartments =

userManagementApiHelperService.getBranchDepartments(workspaceContext.getWorkspa
ceId());
    Map<UUID, BranchDepartmentInfo> branchDepartmentInfoMap =
        branchDepartments.stream().collect(

Collectors.toMap(com.lawcubator.falcon.lms.dtos.pojo.internal.BranchDepartmentI
nfo::getBranchId,
                value -> value, (oldValue, newValue) -> oldValue));

    List<WorkspacePolicyAssignInfo> policyAssignCreateList = new ArrayList<>();
    WorkspacePolicy workspacePolicy = optionalWorkspacePolicy.get();
    if (ObjectUtil.isNotNullOrEmpty(workspaceContext.getRoles()) && (
        workspaceContext.getRoles().contains(WorkspaceUserRole.WORKSPACE_ADMIN)
|| workspaceContext.getRoles()
            .contains(WorkspaceUserRole.EMPLOYER))) {
```

```
        supersedeExistingAssignmentsForWorkspaceLevel(workspacePolicy);

        if (request.isAllBranch()) {
            workspacePolicy.setAssignedToAllBranch(true);
            WorkspacePolicyAssignInfo workspacePolicyAssignInfo =
                getWorkspacePolicyAssignInfo(workspaceContext, null, null,
workspacePolicy);
            workspacePolicyAssignInfo.setForAllBranches(true);
            policyAssignCreateList.add(workspacePolicyAssignInfo);

            /*getWorkspacePolicyAssignInfoListForAllBranches(workspaceContext,
workspacePolicy,
                branchDepartments, policyAssignCreateList);*/
        } else {
            workspacePolicy.setAssignedToAllBranch(true);
            List<BranchInfoRequest> branches = request.getBranches();
            getWorkspacePolicyAssignInfoForRequestBranches(workspaceContext,
workspacePolicy,
                branchDepartmentInfoMap, policyAssignCreateList, branches);
        }
        workspacePolicyService.modify(workspacePolicy);
    } else if (ObjectUtil.isNotNullOrEmpty(workspaceContext.getRoles()) &&
workspaceContext.getRoles()
        .contains(WorkspaceUserRole.BRANCH_ADMIN)) {
        supersedeExistingAssignmentForBranchLevel(workspaceContext,
workspacePolicy);
        // Extract Branch-Department Info only for current user branch.
        List<BranchDepartmentInfo> branchDepartmentInfoList =
            branchDepartments.stream().filter(branch ->
branch.getBranchId().equals(workspaceContext.getBranchId()))
                .toList();
        if (request.isAllBranch()) {
            getWorkspacePolicyAssignInfoListForAllBranches(workspaceContext,
workspacePolicy,
                branchDepartmentInfoList, policyAssignCreateList);
        } else {
            if (ObjectUtil.isNotNullOrEmpty(request.getBranches())) {
                List<BranchInfoRequest> branches =
request.getBranches().stream()
                    .filter(branchRequest -> {
                        if (ObjectUtil.isNull(branchRequest.getBranchId())) {
                            return false;
                        }
                        return
branchRequest.getBranchId().equals(workspaceContext.getBranchId());
                    }).toList();
```

```
                getWorkspacePolicyAssignInfoForRequestBranches(workspaceContext,
workspacePolicy,
                    branchDepartmentInfoMap, policyAssignCreateList, branches);
            }
        }
    }

    if (ObjectUtil.isNotNullOrEmpty(policyAssignCreateList)) {
        workspacePolicyAssignInfoRepository.saveAll(policyAssignCreateList);
    }
}
```

```
private void supersedeExistingAssignmentsForWorkspaceLevel(WorkspacePolicy
workspacePolicy) {
    List<WorkspacePolicyAssignInfo> policyAssignInfoList =
getAllByWorkspacePolicy(workspacePolicy);
    if (ObjectUtil.isNotNullOrEmpty(policyAssignInfoList)) {
        List<WorkspacePolicyAssignInfo> updatedPolicyAssignInfoList =
            policyAssignInfoList.stream().map(policyAssignInfo -> {
                policyAssignInfo.setStatus(RecordStatus.SUPERSEDED);
                return policyAssignInfo;
            }).toList();

workspacePolicyAssignInfoRepository.saveAll(updatedPolicyAssignInfoList);
    }
}
```

```
private void supersedeExistingAssignmentForBranchLevel(WorkspaceContext
workspaceContext,
                                                       WorkspacePolicy
workspacePolicy) {
    List<WorkspacePolicyAssignInfo> workspacePolicyAssignInfoList =
        getByWorkspacePolicyAndWorkspaceIdAndBranchId(workspacePolicy,
workspaceContext.getWorkspaceId(),
            workspaceContext.getBranchId());
    if (ObjectUtil.isNotNullOrEmpty(workspacePolicyAssignInfoList)) {
        List<WorkspacePolicyAssignInfo> updatedPolicyAssignInfoList =
            workspacePolicyAssignInfoList.stream().map(policyAssignInfo -> {
                policyAssignInfo.setStatus(RecordStatus.SUPERSEDED);
                return policyAssignInfo;
            }).toList();

workspacePolicyAssignInfoRepository.saveAll(updatedPolicyAssignInfoList);
    }
}
```

```
private void getWorkspacePolicyAssignInfoForRequestBranches(WorkspaceContext
workspaceContext,
                                                            WorkspacePolicy
workspacePolicy,
                                                            Map<UUID,
BranchDepartmentInfo> branchDepartmentInfoMap,

List<WorkspacePolicyAssignInfo> policyAssignCreateList,

List<BranchInfoRequest> branches) {
   if (ObjectUtil.isNotNullOrEmpty(branches)) {
       for (BranchInfoRequest branch : branches) {
           if (branch.isAllDepartments()) {
               getWorkspacePolicyAssignInfoForAllDepartments(workspaceContext,
                   workspacePolicy, branchDepartmentInfoMap,
policyAssignCreateList, branch);
           } else {
               List<UUID> departmentIds = branch.getDepartmentIds();
               if (ObjectUtil.isNotNullOrEmpty(departmentIds)) {
                   for (UUID departmentId : departmentIds) {
                       WorkspacePolicyAssignInfo workspacePolicyAssignInfo =
                           getWorkspacePolicyAssignInfo(workspaceContext,
branch.getBranchId(),
                               departmentId, workspacePolicy);
                       policyAssignCreateList.add(workspacePolicyAssignInfo);
                   }
               } else {
                   // create record only for branchId
                   WorkspacePolicyAssignInfo workspacePolicyAssignInfo =
                       getWorkspacePolicyAssignInfo(workspaceContext,
branch.getBranchId(), null,
                           workspacePolicy);
                   policyAssignCreateList.add(workspacePolicyAssignInfo);
               }
           }
       }
   }
}
```

```
private void getWorkspacePolicyAssignInfoListForAllBranches(WorkspaceContext
workspaceContext,
                                                            WorkspacePolicy
workspacePolicy,

List<BranchDepartmentInfo> branchDepartmentInfoList,
```

```
List<WorkspacePolicyAssignInfo> policyAssignCreateList) {
   for (BranchDepartmentInfo branchDepartmentInfo : branchDepartmentInfoList) {
       UUID branchId = branchDepartmentInfo.getBranchId();

       List<DepartmentResponse> departments =
branchDepartmentInfo.getDepartments();
       if (ObjectUtil.isNotNullOrEmpty(departments)) {
           for (DepartmentResponse department : departments) {
               WorkspacePolicyAssignInfo workspacePolicyAssignInfo =
                   getWorkspacePolicyAssignInfo(workspaceContext, branchId,
department.getDepartmentId(),
                       workspacePolicy);
               policyAssignCreateList.add(workspacePolicyAssignInfo);
           }
       } else {
           // Create the record for branch only
           WorkspacePolicyAssignInfo workspacePolicyAssignInfo =
               getWorkspacePolicyAssignInfo(workspaceContext, branchId, null,
workspacePolicy);
           policyAssignCreateList.add(workspacePolicyAssignInfo);
       }
   }
}
```

```
@Override
public List<WorkspacePolicyAssignInfo> getAllByWorkspacePolicy(WorkspacePolicy
workspacePolicy) {
   return workspacePolicyAssignInfoRepository
       .findByWorkspacePolicyAndStatus(workspacePolicy, RecordStatus.ACTIVE);
}
```

```
private List<WorkspacePolicyAssignInfo>
getByWorkspacePolicyAndWorkspaceIdAndBranchId(
   WorkspacePolicy workspacePolicy, UUID workspaceId, UUID branchId) {
   return
workspacePolicyAssignInfoRepository.findByWorkspacePolicyAndWorkspaceIdAndBranc
hIdAndStatus(
       workspacePolicy, workspaceId, branchId, RecordStatus.ACTIVE);
}
```

```
private void getWorkspacePolicyAssignInfoForAllDepartments(WorkspaceContext
workspaceContext,
                                                           WorkspacePolicy
workspacePolicy,
```

```
                                                        Map<UUID,
BranchDepartmentInfo> branchDepartmentInfoMap,

List<WorkspacePolicyAssignInfo> policyAssignCreateList,
                                                        BranchInfoRequest
branch) {
   BranchDepartmentInfo branchDepartmentInfo =
branchDepartmentInfoMap.get(branch.getBranchId());
   if (ObjectUtil.isNotNull(branchDepartmentInfo)) {
       List<DepartmentResponse> departments =
branchDepartmentInfo.getDepartments();
       if (ObjectUtil.isNotNullOrEmpty(departments)) {
           for (DepartmentResponse department : departments) {
               WorkspacePolicyAssignInfo workspacePolicyAssignInfo =
                   getWorkspacePolicyAssignInfo(workspaceContext,
branchDepartmentInfo.getBranchId(),
                       department.getDepartmentId(), workspacePolicy);
               policyAssignCreateList.add(workspacePolicyAssignInfo);
           }
       } else {
           // create record only for branch
           WorkspacePolicyAssignInfo workspacePolicyAssignInfo =
               getWorkspacePolicyAssignInfo(workspaceContext,
branchDepartmentInfo.getBranchId(), null,
                   workspacePolicy);
           policyAssignCreateList.add(workspacePolicyAssignInfo);
       }
   }
}
```

```
private WorkspacePolicyAssignInfo getWorkspacePolicyAssignInfo(WorkspaceContext
workspaceContext,
                                                               UUID branchId,
                                                               UUID
departmentId,
                                                               WorkspacePolicy
workspacePolicy) {
   WorkspacePolicyAssignInfo workspacePolicyAssignInfo = new
WorkspacePolicyAssignInfo();

workspacePolicyAssignInfo.setAssignedBy(UUID.fromString(workspaceContext.getUse
rId()));
   workspacePolicyAssignInfo.setAssignedOn(Instant.now());
   workspacePolicyAssignInfo.setWorkspacePolicy(workspacePolicy);
   workspacePolicyAssignInfo.setBranchId(branchId);
   workspacePolicyAssignInfo.setWorkspaceId(workspaceContext.getWorkspaceId());
```

```
    workspacePolicyAssignInfo.setCreatedBy(workspaceContext.getUserId());
    workspacePolicyAssignInfo.setCreatedDate(Instant.now());
    workspacePolicyAssignInfo.setDepartmentId(departmentId);
    return workspacePolicyAssignInfo;
}
```

PolicyController :-

```
@PostMapping("/{policyId}/autoAssign")
@Operation(summary = "API to update the auto assign of branches and Departments
for Policy")
@Authorize(roles = {WorkspaceUserRole.WORKSPACE_ADMIN,
WorkspaceUserRole.EMPLOYER, WorkspaceUserRole.BRANCH_ADMIN})
public ResponseEntity updateAutoAssignBranches(@PathVariable UUID policyId,
                                               @RequestBody
                                               AutoAssignBranchDepartmentRequest
request) {
    workspacePolicyService.updateAutoAssignCapability(getUserTenantContext(),
policyId, request);
    return ResponseEntity.ok().build();
}
```

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.lawcubator.falcon.infra.resource;

import com.lawcubator.falcon.infra.annotations.processors.ReflectionService;
import com.lawcubator.falcon.infra.dto.DTO;
import com.lawcubator.falcon.infra.enums.Environment;
import com.lawcubator.falcon.infra.exceptions.ExceptionGenerator;
import com.lawcubator.falcon.infra.model.IModel;
import com.lawcubator.falcon.infra.model.WorkspaceUserRole;
import com.lawcubator.falcon.infra.util.ObjectUtil;
import com.lawcubator.falcon.infra.workspace.InternalWorkspaceService;
import com.lawcubator.falcon.infra.workspace.dto.WorkspaceContext;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Objects;
import java.util.Set;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Component;
```

```java
@Component
public class Resource<T extends IModel, K extends DTO> extends AbstractResource
{
   @Autowired
   protected ExceptionGenerator exceptionGenerator;
   @Autowired
   protected WorkspaceContextProvider workspaceContextProvider;
   @Autowired
   protected InternalWorkspaceService internalWorkspaceService;

   public Resource() {
   }

   public K getDTO(T model) {
       try {
           return (K)(ReflectionService.getDto(model));
       } catch (Exception e) {
           e.printStackTrace();
           throw
this.exceptionGenerator.generateServerException("OBJECT_CONVERSION_FAILED");
       }
   }

   public T getModel(K dto) {
       try {
           return (T)(ReflectionService.getModel(dto));
       } catch (Exception var3) {
           throw
this.exceptionGenerator.generateServerException("OBJECT_CONVERSION_FAILED");
       }
   }

   public List<K> getDTOList(List<T> modelList) {
       List<K> list = new ArrayList();

       for(T model : modelList) {
           list.add(this.getDTO(model));
       }

       return list;
   }

   public Set<K> getDTOList(Set<T> modelList) {
       Set<K> set = new HashSet();

       for(T model : modelList) {
           set.add(this.getDTO(model));
```

```
        }

        return set;
    }

    protected String getHeader(String pool) {
        return this.req.getHeader(pool);
    }

    public WorkspaceContext getUserTenantContext() {
        String originUrl = this.getHeader("originUrl");
        return this.workspaceContextProvider.getWorkspaceContext(originUrl);
    }

    protected boolean isLawcubatorAdmin() {
        return
this.internalWorkspaceService.isLawcubatorAdmin(this.getUserId());
    }

    public WorkspaceContext authorizeAndPopulate() {
        if (this.isLawcubatorAdmin()) {
            WorkspaceContext workspaceContext =
(WorkspaceContext)SecurityContextHolder.getContext().getAuthentication().getCre
dentials();
            List<WorkspaceUserRole> userRoles =
this.internalWorkspaceService.getUserRoles(this.getUserId());
            if (Objects.isNull(userRoles)) {
                userRoles = new ArrayList();
                userRoles.add(WorkspaceUserRole.LAWCUBATOR_ADMIN);
            }

            workspaceContext.setRoles(userRoles);
            return workspaceContext;
        } else {
            throw
this.exceptionGenerator.generateUnAuthenticatedRuntimeException();
        }
    }

    protected Environment getEnvironment() {
        String originUrl = this.req.getHeader("originUrl");
        if (ObjectUtil.isStringEmptyOrNull(originUrl)) {
            originUrl = this.getHeader("referrer");
        }

        if (!originUrl.endsWith("lawcubator.in") &&
!originUrl.endsWith("lawcubator.com")) {
```

```
            return originUrl.endsWith("lawcubate.com") ? Environment.DEV :
Environment.TEST;
        } else {
            return Environment.PROD;
        }
    }
}
```