```java
package com.example.micko.controller;

import com.example.micko.rule.RuleCacheManager;
import com.example.micko.rule.RuleExecutor;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.tags.Tag;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.Map;

@RestController
@RequestMapping("/api/rules")
@Tag(name = "Rules API", description = "Manage and Execute YAML
        Rules")
public class RulesApiController {

    private final RuleCacheManager cacheManager;
    private final RuleExecutor ruleExecutor;

    public RulesApiController(RuleCacheManager cacheManager,
        RuleExecutor ruleExecutor) {
        this.cacheManager = cacheManager;
        this.ruleExecutor = ruleExecutor;
    }

    /**
     * Execute a Rule by Table Name, Action, and Key (GET)
     */
    @Operation(summary = "Fetch Config by Key", description =
        "Fetch configuration from the table_rule by key")
    @GetMapping("/{tableName}/{action}/{key}")
    public ResponseEntity<Object> executeRuleWithGet(
            @PathVariable String tableName,
            @PathVariable String action,
            @PathVariable String key) {

        // Fetch Rule from Cache
        Map<String, Object> rule =
        cacheManager.getRule(tableName, action);

        if (rule == null) {
            return ResponseEntity.status(404).body("Rule not
        found");
```

```
        }

        // Create Inputs from URL Path Variable
        Map<String, String> params = Map.of("key", key);

        // Execute Rule Logic
        return ruleExecutor.execute(rule, params);
    }
}
```

```
tables:
  - name: "table_rule"
    columns:
      - name: "key"
        type: "string"
      - name: "value"
        type: "string"
    data:
      - key: "APP_MODE"
        value: "PRODUCTION"
      - key: "MAX_RETRIES"
        value: "5"
    rules:
      - action: "fetchConfigByKey"
        validations:
          - type: "query"
            query: "SELECT COUNT(*) FROM table_rule WHERE key
        = :key"
            expectedResult: "1"
            error:
              statusCode: 404
              message: "Configuration key not found."
        executionQuery:
          query: "SELECT * FROM table_rule WHERE key = :key"
          successMessage: "Configuration fetched successfully."
          errorMessage: "Failed to fetch configuration."
```

```
tables:
  - name: "user_table"
    columns:
      - name: "userid"
        type: "string"
      - name: "password"
        type: "string"
```

```
    secure: true
  - name: "deviceid"
    type: "string"
  - name: "firsttimelogin"
    type: "boolean"
  - name: "termandcondition"
    type: "boolean"
  - name: "pushnotificationenabled"
    type: "boolean"

data:
  - userid: "bankuser"
    password: "v1sWejjIQ8UYCAvGI7Yjuw=="
    deviceid: "1a9c47c9-4776-4416-8458-a3c746fa130e"
    firsttimelogin: true
    pushnotificationenabled: true
    termandcondition: true

rules:
  - action: "validateLogin"
    validations:
      - name: "checkLoginWithPassword"
        type: "query"
        query: "SELECT COUNT(*) FROM user_table WHERE userid
    = :userid AND password = :password"
        expectedResult: "1"
        error:
          statusCode: 401
          message: "Invalid username or password."

      - name: "checkLoginWithoutPassword"
        type: "query"
        query: "SELECT COUNT(*) FROM user_table WHERE userid
    = :userid AND firsttimelogin = false"
        expectedResult: "1"
        error:
          statusCode: 400
          message: "User must provide a password on first-
    time login."

    executionQuery:
      query: |
        SELECT userid, deviceid, firsttimelogin,
    termandcondition, pushnotificationenabled
        FROM user_table
```

```
    WHERE userid = :userid
      AND (
        (:password IS NOT NULL AND password = :password)
OR
        (:password IS NULL AND firsttimelogin = false)
      )
  successMessage: "Login successful."
  errorMessage: "Login failed due to unexpected error."
```