attendance.go

```go
package fuzzifikasi

func FuzzifyAttendance(attendance float64) (low, medium, high float64) {
    // Low: <= 0.65 (Poor)
    if attendance <= 0.60 {
        low = 1.0
    } else if attendance <= 0.65 {
        low = (0.65 - attendance) / (0.65 - 0.60)
    }

    // Medium: 0.60-0.85 (Needs Improvement, Satisfactory)
    if attendance >= 0.60 && attendance <= 0.75 {
        medium = (attendance - 0.60) / (0.75 - 0.60)
    } else if attendance > 0.75 && attendance <= 0.85 {
        medium = (0.85 - attendance) / (0.85 - 0.75)
    }

    // High: >= 0.80 (Good, Excellent)
    if attendance >= 0.90 {
        high = 1.0
    } else if attendance >= 0.80 {
        high = (attendance - 0.80) / (0.90 - 0.80)
    }

    return low, medium, high
}
```

cca.go

```go
package fuzzifikasi

func FuzzifyCCA(cca float64) (low, medium, high float64) {
    // Low: <= 55 (Poor)
    if cca <= 50 {
        low = 1.0
    } else if cca <= 55 {
        low = (55 - cca) / (55 - 50)
    }

    // Medium: 50-75 (Needs Improvement, Satisfactory)
    if cca >= 50 && cca <= 65 {
        medium = (cca - 50) / (65 - 50)
    } else if cca > 65 && cca <= 75 {
        medium = (75 - cca) / (75 - 65)
    }

    // High: >= 70 (Good, Excellent)
    if cca >= 80 {
        high = 1.0
    } else if cca >= 70 {
        high = (cca - 70) / (80 - 70)
    }

    return low, medium, high
}
```

final_exam.go

```go
package fuzzifikasi

func FuzzifyFinalExam(finalExam float64) (low, medium, high float64) {
    // Low: <= 54 (Poor)
    if finalExam <= 52 {
        low = 1.0
    } else if finalExam <= 54 {
        low = (54 - finalExam) / (54 - 52)
    }

    // Medium: 52-82 (Needs Improvement, Satisfactory)
    if finalExam >= 52 && finalExam <= 70 {
        medium = (finalExam - 52) / (70 - 52)
    } else if finalExam > 70 && finalExam <= 82 {
        medium = (82 - finalExam) / (82 - 70)
    }

    // High: >= 78 (Good, Excellent)
    if finalExam >= 82 {
        high = 1.0
    } else if finalExam >= 78 {
        high = (finalExam - 78) / (82 - 78)
    }

    return low, medium, high
}
```

gpa.go

```go
package fuzzifikasi

func FuzzifyGPA(gpa float64) (low, medium, high float64) {
    // Low: <= 2.2 (Poor)
    if gpa <= 1.8 {
        low = 1.0
    } else if gpa <= 2.2 {
        low = (2.2 - gpa) / (2.2 - 1.8)
    }

    // Medium: 1.8-3.2 (Needs Improvement, Satisfactory) - DIPERLUAS
    if gpa >= 1.8 && gpa <= 2.5 {
        medium = (gpa - 1.8) / (2.5 - 1.8)
    } else if gpa > 2.5 && gpa <= 3.2 {
        medium = (3.2 - gpa) / (3.2 - 2.5)
    }

    // High: >= 2.8 (Good, Excellent) - OVERLAP DENGAN MEDIUM
    if gpa >= 3.2 {
        high = 1.0
    } else if gpa >= 2.8 {
        high = (gpa - 2.8) / (3.2 - 2.8)
    }

    return low, medium, high
}
```

midterm_exam.go

```go
package fuzzifikasi

func FuzzifyMES(midterm float64) (low, medium, high float64) {
    if midterm <= 55 {
        low = 1.0
    } else if midterm <= 60 {
        low = (60 - midterm) / (60 - 55)
    }

    // Medium: 55-75 (Needs Improvement, Satisfactory)
    if midterm >= 55 && midterm <= 65 {
        medium = (midterm - 55) / (65 - 55)
    } else if midterm > 65 && midterm <= 75 {
        medium = (75 - midterm) / (75 - 65)
    }

    // High: >= 70 (Good, Excellent)
    if midterm >= 80 {
        high = 1.0
    } else if midterm >= 70 {
        high = (midterm - 70) / (80 - 70)
    }

    return low, medium, high
}
```

rules.go

```go
package inferensi

type Rule struct {
    GPA         string
    CCA         string
    Attendance  string
    MidtermExam string
    FinalExam   string
    Performance string
}

func Rules() []Rule {
    return []Rule{
        // ===== POOR RULES =====
             {GPA: "Low", CCA: "Low", Attendance: "Low", MidtermExam: "Low", FinalExam: "Low",
Performance: "Poor"},
           {GPA: "Low", CCA: "Low", Attendance: "Low", MidtermExam: "Low", FinalExam: "Medium",
Performance: "Poor"},
           {GPA: "Low", CCA: "Low", Attendance: "Low", MidtermExam: "Medium", FinalExam: "Low",
Performance: "Poor"},
           {GPA: "Low", CCA: "Low", Attendance: "Medium", MidtermExam: "Low", FinalExam: "Low",
Performance: "Poor"},
           {GPA: "Low", CCA: "Medium", Attendance: "Low", MidtermExam: "Low", FinalExam: "Low",
Performance: "Poor"},
           {GPA: "Medium", CCA: "Low", Attendance: "Low", MidtermExam: "Low", FinalExam: "Low",
Performance: "Poor"},

        // ===== NEEDS IMPROVEMENT RULES - EXPANDED =====
        // Kasus dengan mayoritas Medium tapi ada Low values (seperti kasus student 4)
            {GPA: "Medium", CCA: "Medium", Attendance: "Medium", MidtermExam: "Low", FinalExam:
"Medium", Performance: "Needs Improvement"},
          {GPA: "Medium", CCA: "Medium", Attendance: "Medium", MidtermExam: "Medium", FinalExam:
"Low", Performance: "Needs Improvement"},
            {GPA: "Medium", CCA: "Medium", Attendance: "Low", MidtermExam: "Medium", FinalExam:
"Medium", Performance: "Needs Improvement"},
            {GPA: "Medium", CCA: "Low", Attendance: "Medium", MidtermExam: "Medium", FinalExam:
"Medium", Performance: "Needs Improvement"},
            {GPA: "Low", CCA: "Medium", Attendance: "Medium", MidtermExam: "Medium", FinalExam:
"Medium", Performance: "Needs Improvement"},

        // Kasus dengan campuran Low dan Medium
        {GPA: "Low", CCA: "Low", Attendance: "Medium", MidtermExam: "Medium", FinalExam: "Medium",
Performance: "Needs Improvement"},
        {GPA: "Low", CCA: "Medium", Attendance: "Low", MidtermExam: "Medium", FinalExam: "Medium",
Performance: "Needs Improvement"},
        {GPA: "Low", CCA: "Medium", Attendance: "Medium", MidtermExam: "Low", FinalExam: "Medium",
Performance: "Needs Improvement"},
        {GPA: "Low", CCA: "Medium", Attendance: "Medium", MidtermExam: "Medium", FinalExam: "Low",
Performance: "Needs Improvement"},
        {GPA: "Medium", CCA: "Low", Attendance: "Low", MidtermExam: "Medium", FinalExam: "Medium",
Performance: "Needs Improvement"},
        {GPA: "Medium", CCA: "Low", Attendance: "Medium", MidtermExam: "Low", FinalExam: "Medium",
Performance: "Needs Improvement"},
        {GPA: "Medium", CCA: "Low", Attendance: "Medium", MidtermExam: "Medium", FinalExam: "Low",
Performance: "Needs Improvement"},
        {GPA: "Medium", CCA: "Medium", Attendance: "Low", MidtermExam: "Low", FinalExam: "Medium",
Performance: "Needs Improvement"},
        {GPA: "Medium", CCA: "Medium", Attendance: "Low", MidtermExam: "Medium", FinalExam: "Low",
Performance: "Needs Improvement"},
        {GPA: "Medium", CCA: "Medium", Attendance: "Medium", MidtermExam: "Low", FinalExam: "Low",
Performance: "Needs Improvement"},

        // Kasus dengan 2+ Low values
         {GPA: "Low", CCA: "Low", Attendance: "High", MidtermExam: "Medium", FinalExam: "Medium",
Performance: "Needs Improvement"},
          {GPA: "Low", CCA: "Medium", Attendance: "Low", MidtermExam: "Low", FinalExam: "Medium",
Performance: "Needs Improvement"},
          {GPA: "Medium", CCA: "Low", Attendance: "Low", MidtermExam: "Low", FinalExam: "Medium",
Performance: "Needs Improvement"},

        // ===== SATISFACTORY RULES - HANYA UNTUK KASUS MURNI MEDIUM =====
        // Hanya kasus dengan semua atau mayoritas Medium tanpa Low values
          {GPA: "Medium", CCA: "Medium", Attendance: "Medium", MidtermExam: "Medium", FinalExam:
"Medium", Performance: "Satisfactory"},

        // Kasus dengan 1 High value tapi sisanya Medium (tidak ada Low)
           {GPA: "High", CCA: "Medium", Attendance: "Medium", MidtermExam: "Medium", FinalExam:
"Medium", Performance: "Satisfactory"},
           {GPA: "Medium", CCA: "High", Attendance: "Medium", MidtermExam: "Medium", FinalExam:
"Medium", Performance: "Satisfactory"},
           {GPA: "Medium", CCA: "Medium", Attendance: "High", MidtermExam: "Medium", FinalExam:
"Medium", Performance: "Satisfactory"},
           {GPA: "Medium", CCA: "Medium", Attendance: "Medium", MidtermExam: "High", FinalExam:
"Medium", Performance: "Satisfactory"},
          {GPA: "Medium", CCA: "Medium", Attendance: "Medium", MidtermExam: "Medium", FinalExam:
"High", Performance: "Satisfactory"},

        // Kasus dengan 2 High values
            {GPA: "High", CCA: "High", Attendance: "Medium", MidtermExam: "Medium", FinalExam:
"Medium", Performance: "Satisfactory"},
            {GPA: "High", CCA: "Medium", Attendance: "High", MidtermExam: "Medium", FinalExam:
"Medium", Performance: "Satisfactory"},
            {GPA: "High", CCA: "Medium", Attendance: "Medium", MidtermExam: "High", FinalExam:
"Medium", Performance: "Satisfactory"},
           {GPA: "High", CCA: "Medium", Attendance: "Medium", MidtermExam: "Medium", FinalExam:
"High", Performance: "Satisfactory"},
            {GPA: "Medium", CCA: "High", Attendance: "High", MidtermExam: "Medium", FinalExam:
"Medium", Performance: "Satisfactory"},
            {GPA: "Medium", CCA: "High", Attendance: "Medium", MidtermExam: "High", FinalExam:
"Medium", Performance: "Satisfactory"},
           {GPA: "Medium", CCA: "High", Attendance: "Medium", MidtermExam: "Medium", FinalExam:
"High", Performance: "Satisfactory"},
            {GPA: "Medium", CCA: "Medium", Attendance: "High", MidtermExam: "High", FinalExam:
"Medium", Performance: "Satisfactory"},
           {GPA: "Medium", CCA: "Medium", Attendance: "High", MidtermExam: "Medium", FinalExam:
"High", Performance: "Satisfactory"},
           {GPA: "Medium", CCA: "Medium", Attendance: "Medium", MidtermExam: "High", FinalExam: "High", Performance: "Satisfactory"},

        // ===== GOOD RULES =====
        // Untuk kasus dengan mayoritas High (3+ High values)
        {GPA: "High", CCA: "High", Attendance: "High", MidtermExam: "Medium", FinalExam: "Medium", Performance: "Good"},
        {GPA: "High", CCA: "High", Attendance: "Medium", MidtermExam: "High", FinalExam: "Medium", Performance: "Good"},
        {GPA: "High", CCA: "High", Attendance: "Medium", MidtermExam: "Medium", FinalExam: "High", Performance: "Good"},
        {GPA: "High", CCA: "Medium", Attendance: "High", MidtermExam: "High", FinalExam: "Medium", Performance: "Good"},
        {GPA: "High", CCA: "Medium", Attendance: "High", MidtermExam: "Medium", FinalExam: "High", Performance: "Good"},
        {GPA: "High", CCA: "Medium", Attendance: "Medium", MidtermExam: "High", FinalExam: "High", Performance: "Good"},
        {GPA: "Medium", CCA: "High", Attendance: "High", MidtermExam: "High", FinalExam: "Medium", Performance: "Good"},
        {GPA: "Medium", CCA: "High", Attendance: "High", MidtermExam: "Medium", FinalExam: "High", Performance: "Good"},
        {GPA: "Medium", CCA: "High", Attendance: "Medium", MidtermExam: "High", FinalExam: "High", Performance: "Good"},
        {GPA: "Medium", CCA: "Medium", Attendance: "High", MidtermExam: "High", FinalExam: "High", Performance: "Good"},

        // Kasus dengan 4+ High values
         {GPA: "High", CCA: "High", Attendance: "High", MidtermExam: "High", FinalExam: "Medium", Performance: "Good"},
         {GPA: "High", CCA: "High", Attendance: "High", MidtermExam: "Medium", FinalExam: "High", Performance: "Good"},
         {GPA: "High", CCA: "High", Attendance: "Medium", MidtermExam: "High", FinalExam: "High", Performance: "Good"},
         {GPA: "High", CCA: "Medium", Attendance: "High", MidtermExam: "High", FinalExam: "High", Performance: "Good"},
         {GPA: "Medium", CCA: "High", Attendance: "High", MidtermExam: "High", FinalExam: "High", Performance: "Good"},

        // ===== EXCELLENT RULES =====
          {GPA: "High", CCA: "High", Attendance: "High", MidtermExam: "High", FinalExam: "High", Performance: "Excellent"},
         {GPA: "High", CCA: "Medium", Attendance: "High", MidtermExam: "High", FinalExam: "High", Performance: "Excellent"},
         {GPA: "High", CCA: "High", Attendance: "Medium", MidtermExam: "High", FinalExam: "High", Performance: "Excellent"},
    }
}
```

inferensi.go

```go
package inferensi

import (
    "math"
    "tsukamoto/internal/modules/fuzzifikasi"
)

// TsukamotoResult represents the result from Tsukamoto inference
type TsukamotoResult struct {
    WeightedSum float64
    TotalWeight float64
    CrispOutput float64
    RuleOutputs []RuleOutput
}

// RuleOutput represents individual rule calculation
type RuleOutput struct {
    RuleIndex      int
    FiringStrength float64
    CrispValue     float64
    WeightedValue  float64
    Performance    string
}

// Performance value mapping for Tsukamoto (crisp values)
var performanceValues = map[string]float64{
    "Poor":              20.0, // 0-40
    "Needs Improvement": 50.0, // 30-60
    "Satisfactory":      70.0, // 60-80
    "Good":              85.0, // 75-95
    "Excellent":         95.0, // 90-100
}

func TsukamotoInference(gpa, cca, attendance, midterm, finalExam float64) TsukamotoResult {
    // Fuzzify inputs
    gpaLow, gpaMedium, gpaHigh := fuzzifikasi.FuzzifyGPA(gpa)
    ccaLow, ccaMedium, ccaHigh := fuzzifikasi.FuzzifyCCA(cca)
    attLow, attMedium, attHigh := fuzzifikasi.FuzzifyAttendance(attendance)
    midLow, midMedium, midHigh := fuzzifikasi.FuzzifyMES(midterm)
    finLow, finMedium, finHigh := fuzzifikasi.FuzzifyFinalExam(finalExam)

    // Initialize output membership map
    var weightedSum, totalWeight float64
    var ruleOutputs []RuleOutput

    // Get rules
    rules := Rules()

    // Apply each rule using Tsukamoto method
    for i, rule := range rules {
        // Get membership values for inputs based on rule conditions
        var gpaMem, ccaMem, attMem, midMem, finMem float64
        switch rule.GPA {
        case "Low":
            gpaMem = gpaLow
        case "Medium":
            gpaMem = gpaMedium
        case "High":
            gpaMem = gpaHigh
        }

        switch rule.CCA {
        case "Low":
            ccaMem = ccaLow
        case "Medium":
            ccaMem = ccaMedium
        case "High":
            ccaMem = ccaHigh
        }

        switch rule.Attendance {
        case "Low":
            attMem = attLow
        case "Medium":
            attMem = attMedium
        case "High":
            attMem = attHigh
        }

        switch rule.MidtermExam {
        case "Low":
            midMem = midLow
        case "Medium":
            midMem = midMedium
        case "High":
            midMem = midHigh
        }

        switch rule.FinalExam {
        case "Low":
            finMem = finLow
        case "Medium":
            finMem = finMedium
        case "High":
            finMem = finHigh
        }

        // Calculate rule firing strength using minimum (AND operation)
         firingStrength := math.Min(math.Min(math.Min(math.Min(gpaMem, ccaMem), attMem), midMem),
finMem)
        if firingStrength <= 0 {
            continue
        }
        crispValue := performanceValues[rule.Performance]
        weightedValue := firingStrength * crispValue
```

```go
        weightedSum += weightedValue
        totalWeight += firingStrength
        ruleOutputs = append(ruleOutputs, RuleOutput{
            RuleIndex:      i,
            FiringStrength: firingStrength,
            CrispValue:     crispValue,
            WeightedValue:  weightedValue,
            Performance:    rule.Performance,
        })
    }
    var crispOutput float64
    if totalWeight > 0 {
        crispOutput = weightedSum / totalWeight
    }
    return TsukamotoResult{
        WeightedSum: weightedSum,
        TotalWeight: totalWeight,
        CrispOutput: crispOutput,
        RuleOutputs: ruleOutputs,
    }
}

// Legacy function for backward compatibility (converts to old format)
func Inference(gpa, cca, attendance, midterm, finalExam float64) map[string]float64 {
    result := TsukamotoInference(gpa, cca, attendance, midterm, finalExam)
    output := map[string]float64{
        "Poor":              0.0,
        "Needs Improvement": 0.0,
        "Satisfactory":      0.0,
        "Good":              0.0,
        "Excellent":         0.0,
    }
    crispValue := result.CrispOutput
    if crispValue <= 40 {
        output["Poor"] = 1.0
    } else if crispValue <= 60 {
        output["Needs Improvement"] = 1.0
    } else if crispValue <= 80 {
        output["Satisfactory"] = 1.0
    } else if crispValue <= 95 {
        output["Good"] = 1.0
    } else {
        output["Excellent"] = 1.0
    }
    return output
}
```

deffuzifikasi.go

```go
package deffuzifikasi

import (
    "errors"
    "fmt"
    "tsukamoto/internal/modules/inferensi"
)

// TsukamotoDefuzzify performs defuzzification using Tsukamoto method
func TsukamotoDefuzzify(gpa, cca, attendance, midterm, finalExam float64) (string, float64, error)
{
    result := inferensi.TsukamotoInference(gpa, cca, attendance, midterm, finalExam)
    if result.TotalWeight == 0 {
        return "", 0, errors.New("no rule activated: all firing strengths are zero")
    }
    crispOutput := result.CrispOutput
    var category string
    if crispOutput <= 40 {
        category = "Poor"
    } else if crispOutput <= 60 {
        category = "Needs Improvement"
    } else if crispOutput <= 80 {
        category = "Satisfactory"
    } else if crispOutput <= 95 {
        category = "Good"
    } else {
        category = "Excellent"
    }
    return category, crispOutput, nil
}

// Backward compatibility with old Defuzzify function
func Defuzzify(output map[string]float64) (string, error) {
    hasNonZero := false
    for _, membership := range output {
        if membership > 0 {
            hasNonZero = true
            break
        }
    }
    if !hasNonZero {
        return "", errors.New("no rule activated: all membership values are zero")
    }
    var maxMembership float64
    var result string
    for performance, membership := range output {
        if membership > maxMembership {
            maxMembership = membership
            result = performance
        }
    }
    return result, nil
```

```
}

// DefuzzifyStrict - alternative strict defuzzification
func DefuzzifyStrict(output map[string]float64) (string, error) {
    hasNonZero := false
    for _, membership := range output {
        if membership > 0 {
            hasNonZero = true
            break
        }
    }
    if !hasNonZero {
        return "", errors.New("no rule activated: all membership values are zero")
    }
    thresholds := map[string]float64{
        "Poor":              0.3,
        "Needs Improvement": 0.2,
        "Satisfactory":      0.4,
        "Good":              0.5,
        "Excellent":         0.7,
    }
    priorities := map[string]int{
        "Poor":              1,
        "Needs Improvement": 2,
        "Satisfactory":      3,
        "Good":              4,
        "Excellent":         5,
    }
    var candidates []string
    maxMembership := 0.0
    for performance, membership := range output {
        if membership > 0 and membership >= thresholds[performance] {
            if membership > maxMembership {
                maxMembership = membership
                candidates = []string{performance}
            } else if membership == maxMembership {
                candidates = append(candidates, performance)
            }
        }
    }
    if len(candidates) == 0 {
        return Defuzzify(output)
    }
    if len(candidates) == 1 {
        return candidates[0], nil
    }
    result := candidates[0]
    minPriority := priorities[result]
    for _, candidate := range candidates[1:] {
        if priorities[candidate] < minPriority {
            result = candidate
            minPriority = priorities[candidate]
        }
    }
```

```go
    return result, nil
}

// DebugOutput displays inference output for debugging
func DebugOutput(output map[string]float64) {
    fmt.Println("=== INFERENCE OUTPUT DEBUG ===")
    for performance, membership := range output {
        if membership > 0 {
            fmt.Printf("%s: %.6f
", performance, membership)
        }
    }
    fmt.Println("===============================")
}

// DebugTsukamotoOutput displays Tsukamoto inference results for debugging
func DebugTsukamotoOutput(gpa, cca, attendance, midterm, finalExam float64) {
    result := inferensi.TsukamotoInference(gpa, cca, attendance, midterm, finalExam)
    fmt.Println("=== TSUKAMOTO INFERENCE DEBUG ===")
    fmt.Printf("Total Weight: %.6f
", result.TotalWeight)
    fmt.Printf("Weighted Sum: %.6f
", result.WeightedSum)
    fmt.Printf("Crisp Output: %.6f
", result.CrispOutput)
    fmt.Println("\nActive Rules:")
    for _, ruleOutput := range result.RuleOutputs {
        fmt.Printf("Rule %d: %s\n", ruleOutput.RuleIndex, ruleOutput.Performance)
        fmt.Printf("  Firing Strength: %.6f\n", ruleOutput.FiringStrength)
        fmt.Printf("  Crisp Value: %.6f\n", ruleOutput.CrispValue)
        fmt.Printf("  Weighted Value: %.6f\n", ruleOutput.WeightedValue)
        fmt.Println()
    }
    category, crispValue, err := TsukamotoDefuzzify(gpa, cca, attendance, midterm, finalExam)
    if err != nil {
        fmt.Printf("Error: %v\n", err)
    } else {
        fmt.Printf("Final Result: %s (%.2f)\n", category, crispValue)
    }
    fmt.Println("==================================")
}
```

fuzzy/handler.go

```go
package fuzzy

import (
    "net/http"
    "strconv"

    "tsukamoto/internal/models"
    "tsukamoto/internal/modules/deffuzifikasi"
    "tsukamoto/internal/modules/fuzzifikasi"
    "tsukamoto/internal/modules/inferensi"
    "tsukamoto/internal/utils"

    "github.com/gorilla/mux"
    "gorm.io/gorm"
)

// Handler struct holds dependencies for fuzzy handlers
type Handler struct {
    DB *gorm.DB
}

// NewHandler creates a new Handler
func NewHandler(db *gorm.DB) *Handler {
    return &Handler{DB: db}
}

// FuzzyByUserID handles GET /fuzzy/:id
func (h *Handler) FuzzyByUserID(w http.ResponseWriter, r *http.Request) {
    // ...existing code...
}
```

fuzzy/route.go

```go
package fuzzy

import (
    "github.com/gorilla/mux"
    "gorm.io/gorm"
)

// RegisterRoutes registers fuzzy routes
func FuzzyRoute(r *mux.Router, db *gorm.DB) {
    handler := NewHandler(db)
    r.HandleFunc("/fuzzy/{id}", handler.FuzzyByUserID).Methods("GET")
}
```