// Content Generation Service

interface BeigeScore {

total: number;

personalization: number;

engagement: number;

tone: number;

details: {

userStories: number;

audienceMatch: number;

roleRelevance: number;

hookStrength: number;

trendingTopics: number;

uniqueInsights: number;

voiceConsistency: number;

languageQuality: number;

};

}

class ContentGenerationService {

async generateContent(userId: string, type: string): Promise<{

content: string;

score: BeigeScore;

}> {

// Check exercise completion

const exercises = await this.checkExercises(userId);

try {

// Generate content based on exercise completion

const content = await this.getContent(exercises, type);

// Calculate Beige score

const score = await this.calculateBeigeScore(content, exercises);

// Log generation attempt

await this.logGeneration(userId, content, score);

return { content, score };

} catch (error) {

// Handle failure and use fallback

const fallback = await this.getFallbackContent(type);

await this.logError(userId, error);

return fallback;

}

}

private async calculateBeigeScore(content: string, exercises: any): Promise<BeigeScore> {

// Calculate component scores

const personalization = await this.scorePersonalization(content, exercises);

const engagement = await this.scoreEngagement(content);

const tone = await this.scoreTone(content, exercises);

// Calculate weighted total

const total = (

personalization \* 0.4 +

engagement \* 0.35 +

tone \* 0.25

);

return {

total,

personalization,

engagement,

tone,

details: {

// ... detailed scoring breakdown

}

};

}

}