A/B Testing Platform - Low-Level Design (LLD)

Table of Contents

Overview

This Low-Level Design document provides implementation details for the A/B Testing Platform, building upon the High-Level Design. It includes detailed class structures, algorithms, database schemas, and code examples.

Technology Stack

- **Language**: Java (Spring Boot) / Python (FastAPI)
- **Database**: PostgreSQL 14+
- **Cache**: Redis 6+
- **Message Queue**: Apache Kafka
- **Analytics DB**: ClickHouse
- **API Framework**: REST + WebSocket

Class Diagrams & Data Models

Core Domain Models

```
// ============================================
// 1. EXPERIMENT MODEL
// ============================================

public class Experiment {
    // Identifiers
    private UUID id;
    private String key;              // Unique key:
"checkout_redesign_v2"
    private String name;             // Human-readable name
    private String description;

    // Status
    private ExperimentStatus status; // DRAFT, RUNNING, PAUSED,
COMPLETED
```

```
    private LocalDateTime startDate;
    private LocalDateTime endDate;

    // Configuration
    private ExperimentType type;     // AB_TEST, MULTIVARIATE,
FEATURE_FLAG
    private String hypothesis;       // What we're testing
    private MetricDefinition primaryMetric;
    private List<MetricDefinition> secondaryMetrics;

    // Variants
    private List<Variant> variants;
    private TrafficAllocation trafficAllocation;

    // Targeting
    private TargetingRules targetingRules;
    private List<String> excludedUserIds;

    // Metadata
    private UUID createdBy;
    private UUID organizationId;
    private LocalDateTime createdAt;
    private LocalDateTime updatedAt;

    // Business methods
    public Variant assignUser(String userId, UserContext context);
    public boolean isEligible(User user, UserContext context);
    public void validateConfiguration() throws
InvalidExperimentException;
    public ExperimentResults getResults();
}

public enum ExperimentStatus {
    DRAFT,           // Not started, can be modified
    SCHEDULED,       // Scheduled to start
    RUNNING,         // Active, collecting data
    PAUSED,          // Temporarily stopped
    COMPLETED,       // Finished, analyzing results
    ARCHIVED         // Historical, read-only
}

public enum ExperimentType {
    AB_TEST,         // 2 variants
    MULTIVARIATE,    // Multiple variants
    FEATURE_FLAG     // Gradual rollout
}
```

```
// ============================================
// 2. VARIANT MODEL
// ============================================
```

```
public class Variant {
    private UUID id;
    private UUID experimentId;
    private String key;              // "control", "treatment",
"variant_b"
    private String name;             // Human-readable
    private String description;

    // Traffic allocation
    private Integer trafficPercentage; // 0-100
    private boolean isControl;         // Baseline variant

    // Configuration
    private Map<String, Object> config; // Feature-specific config

    // Metadata
    private LocalDateTime createdAt;

    // Methods
    public boolean matchesAllocation(int hashValue);
    public String getConfigValue(String key);
}
```

```
// ============================================
// 3. USER ASSIGNMENT MODEL
// ============================================

public class UserAssignment {
    private UUID id;
    private String userId;
    private UUID experimentId;
    private UUID variantId;

    // Assignment details
    private LocalDateTime assignedAt;
    private AssignmentMethod method;  // HASH, MANUAL, OVERRIDE
    private Integer bucketNumber;     // 0-99 (from hash)

    // Tracking
    private boolean isSticky;         // Once assigned, never change
    private LocalDateTime firstExposure;
    private Integer exposureCount;

    // Methods
    public boolean isActive();
    public Variant getVariant();
    public void recordExposure();
}
```

```
public enum AssignmentMethod {
    HASH,           // Deterministic hash-based
    MANUAL,         // Explicitly assigned
    OVERRIDE,       // QA/testing override
    RESERVED        // Reserved for future use
}
```

```
// ============================================
// 4. EVENT MODEL
// ============================================

public class Event {
    // Core identifiers
    private UUID eventId;
    private String userId;
    private UUID experimentId;
    private UUID variantId;

    // Event details
```