Feature Flag System - Quick Reference

🚀 Quick Start

1. Create a Feature Flag

```
POST /api/feature-flags
{
  "key": "MY_FEATURE",
  "name": "My Awesome Feature",
  "enabled": true,
  "allowedRoles": ["ADMIN", "CUSTOMER"],
  "rolloutPercentage": 100
}
```

2. Protect a Route

```
import { requireFeature } from '../Middlewares/featureFlagMiddleware';

router.get('/my-feature', requireFeature('MY_FEATURE'), controller);
```

3. Get User Features

```
GET /api/features
→ { "MY_FEATURE": true, "OTHER_FEATURE": false }
```

📌 Common Commands

Seed Initial Flags

```
bun run scripts/seedFeatureFlags.ts
```

Check All Flags (Admin)

```
GET /api/feature-flags
```

Enable/Disable Flag

```
PUT /api/feature-flags/MY_FEATURE
{ "enabled": false }
```

Set Rollout to 50%

```
PUT /api/feature-flags/MY_FEATURE
{ "rolloutPercentage": 50 }
```

Clear Cache

```
DELETE /api/feature-flags/cache/clear
```

🎯 Common Scenarios

Scenario 1: Admin-Only Feature

```
{
  "enabled": true,
  "allowedRoles": ["ADMIN"],
  "rolloutPercentage": 100
}
```

Scenario 2: Gradual Rollout to All Users

```
{
  "enabled": true,
  "allowedRoles": ["ADMIN", "CUSTOMER", "PHARMACIST"],
  "rolloutPercentage": 25  // Start with 25%
}
```

Scenario 3: Whitelist Specific Users

```
{
  "enabled": true,
  "allowedUserIds": ["user_id_1", "user_id_2"],
  "rolloutPercentage": 0  // Only whitelisted users
}
```

Scenario 4: Emergency Kill Switch

```
{
  "enabled": false  // Instant disable for everyone
}
```

🔧 Middleware Usage

Basic Protection

```
router.get('/route', requireFeature('FEATURE'), controller);
```

Multiple Protections

```
router.post(
  '/payment',
  customersMiddleware,          // Auth check
  requireFeature('ONLINE_PAYMENT'),  // Feature check
  processPayment
);
```

Programmatic Check

```
const isEnabled = await FeatureFlagService.isFeatureEnabled(
  'FEATURE_KEY',
  userId,
  userRole
);
```

📊 Feature Evaluation Priority

1. **Is feature globally enabled?** → If NO, deny
2. **Is user in whitelist?** → If YES, allow (bypass all)
3. **Is user role allowed?** → If NO, deny
4. **Is user in rollout %?** → If NO, deny
5. **Allow access** ☑

💻 API Endpoints Cheat Sheet

Method	Endpoint	Access	Purpose
POST	`/api/feature-flags`	Admin	Create flag
GET	`/api/feature-flags`	Admin	List all flags
GET	`/api/feature-flags/:key`	Admin	Get one flag
PUT	`/api/feature-flags/:key`	Admin	Update flag
DELETE	`/api/feature-flags/:key`	Admin	Delete flag
POST	`/api/feature-flags/bulk-update`	Admin	Bulk update
DELETE	`/api/feature-flags/cache/clear`	Admin	Clear cache
GET	`/api/features`	User	Get my features

Frontend Integration

React/React Native Example

```
// Fetch enabled features on login
const response = await fetch('/api/features', {
  headers: { Authorization: `Bearer ${token}` }
});
const features = await response.json();

// Use in component
{features.ONLINE_PAYMENT && <PaymentButton />}
{features.AI_CHATBOT && <ChatbotWidget />}
```

Debugging

Check if feature exists

```
GET /api/feature-flags/FEATURE_KEY
```

Check what user sees

```
GET /api/features
```

Clear cache if stale

```
DELETE /api/feature-flags/cache/clear
```

Performance

- Cached: **<1ms response**
- Uncached: **~10ms response**
- Cache TTL: **1 hour**
- Cache invalidation: **On update/delete**

Roles Enum

```
RoleIndex.ADMIN       // Full access
RoleIndex.PHARMACIST  // Pharmacist-specific features
RoleIndex.CUSTOMER    // Regular users
RoleIndex.UNKNOWN     // Google sign-in users
```

File Locations

```
Services/featureFlag.Service.ts          # Business logic
Middlewares/featureFlagMiddleware.ts     # requireFeature()
Routers/Routers/featureFlag.Routes.ts    # Admin APIs
Routers/Routers/features.Routes.ts       # Public API
Databases/Schema/featureFlag.Schema.ts   # MongoDB schema
scripts/seedFeatureFlags.ts              # Seed script
```

Pro Tips

1. **Start with 0% rollout** for new features
2. **Use whitelist** for beta testers
3. **Monitor cache hit ratio** in Redis
4. **Clear cache** after bulk updates
5. **Fail closed** on errors (deny access)
6. **Use consistent naming** (UPPER_SNAKE_CASE)
7. **Document feature** in description field

Example: Gradual Feature Rollout

```
# Day 1: Create disabled
POST { "enabled": false, "rolloutPercentage": 0 }

# Day 2: Enable for admins only
PUT { "enabled": true, "allowedRoles": ["ADMIN"] }

# Day 3: Test with beta users
PUT { "allowedUserIds": ["user1", "user2"] }

# Day 4: 10% rollout to customers
PUT { "allowedRoles": ["ADMIN", "CUSTOMER"], "rolloutPercentage": 10 }

# Week 2: 50% rollout
PUT { "rolloutPercentage": 50 }

# Week 3: 100% rollout
PUT { "rolloutPercentage": 100 }

# Emergency: Disable
PUT { "enabled": false }
```

📞 Support

- Full docs: `FEATURE_FLAG_SYSTEM.md`
- Examples: `examples/featureFlag.examples.ts`
- Seed script: `scripts/seedFeatureFlags.ts`