

InterEd Admin Portal Security Architecture

Role-Based Access Control (RBAC) Implementation

Super Admin

Complete system access

Regional Manager

Full regional access

Counselor

Student-facing operations

University Relations

Institution management

Finance Team

Payment and financial access

Permission Hierarchy

Resource-level

Action-level

Context-aware

Time-based access

Data Encryption Strategy

At Rest

Stored data

In Transit

HTTPS/TLS 1.2+

JWT Tokens

Secure auth

API Security Measures

Rate Limiting

600 req/min

Input Validation

All endpoints

Security Headers

HSTS, CSP, etc.

Audit Logging Implementation

Authentication Events

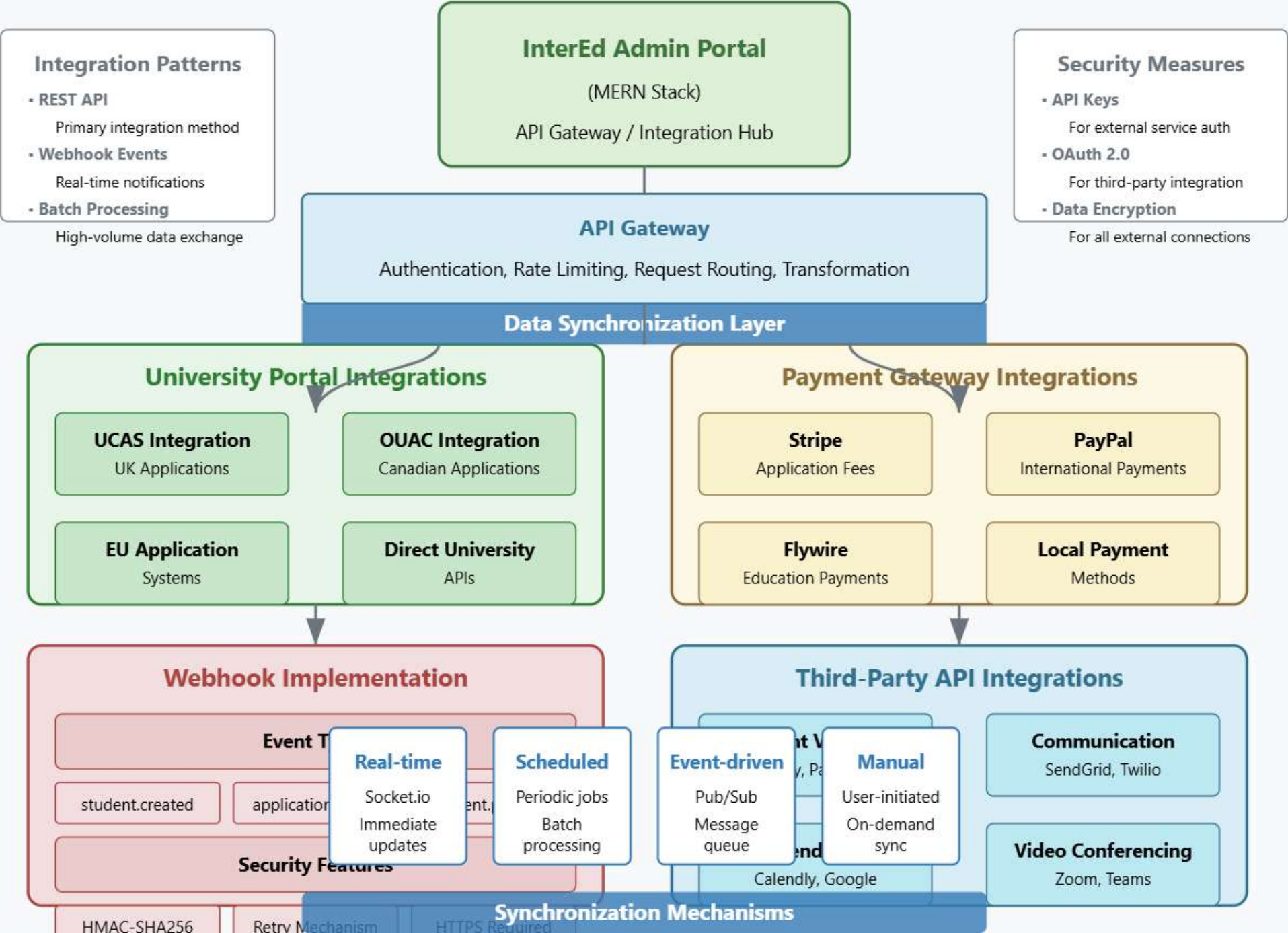
Data Modifications

Admin Actions

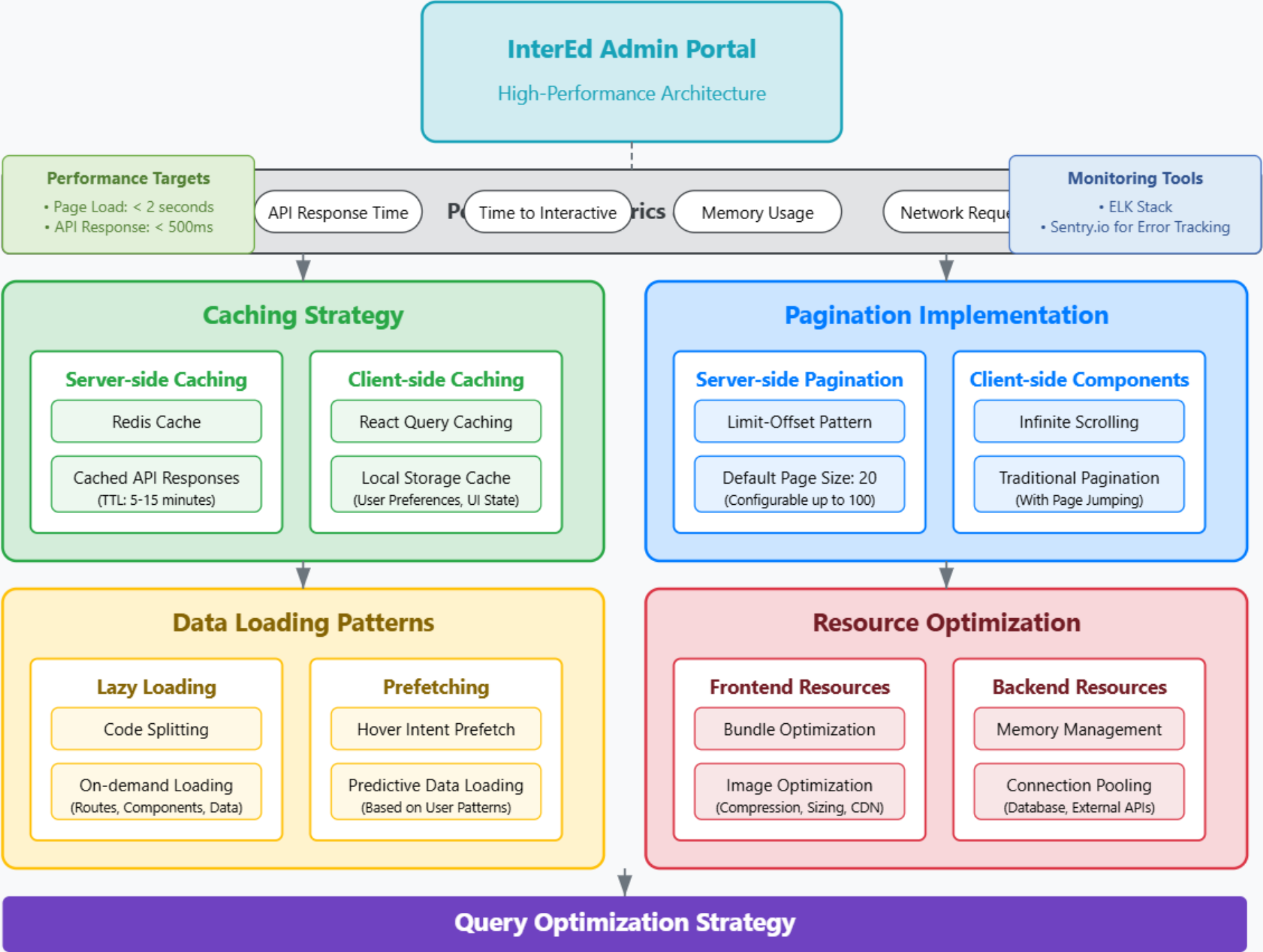
System Access Logs

Error Tracking

InterEd Admin Portal Integration Architecture



InterEd Admin Portal Performance Optimization Strategy

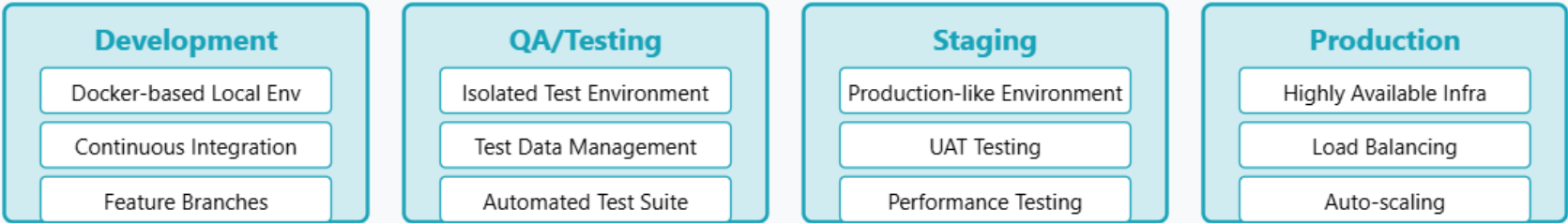


InterEd Admin Portal DevOps Pipeline

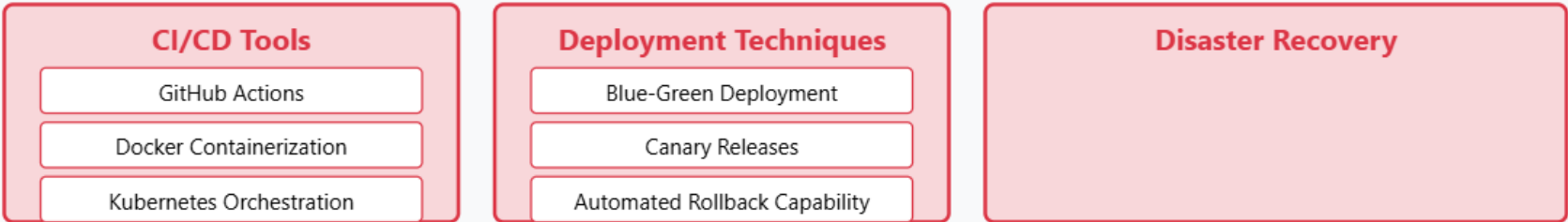
Continuous Integration / Continuous Deployment Workflow



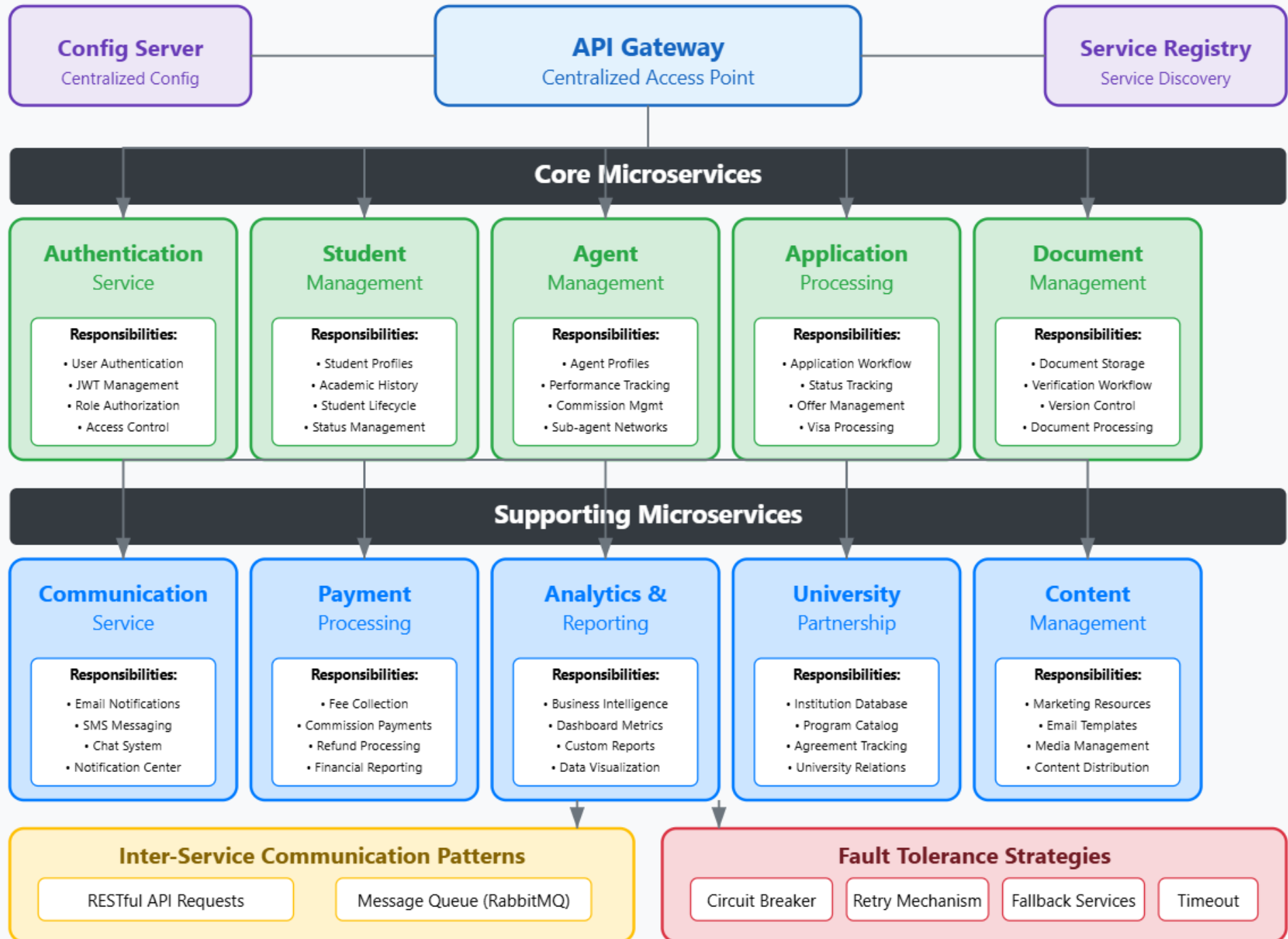
Environment Configuration



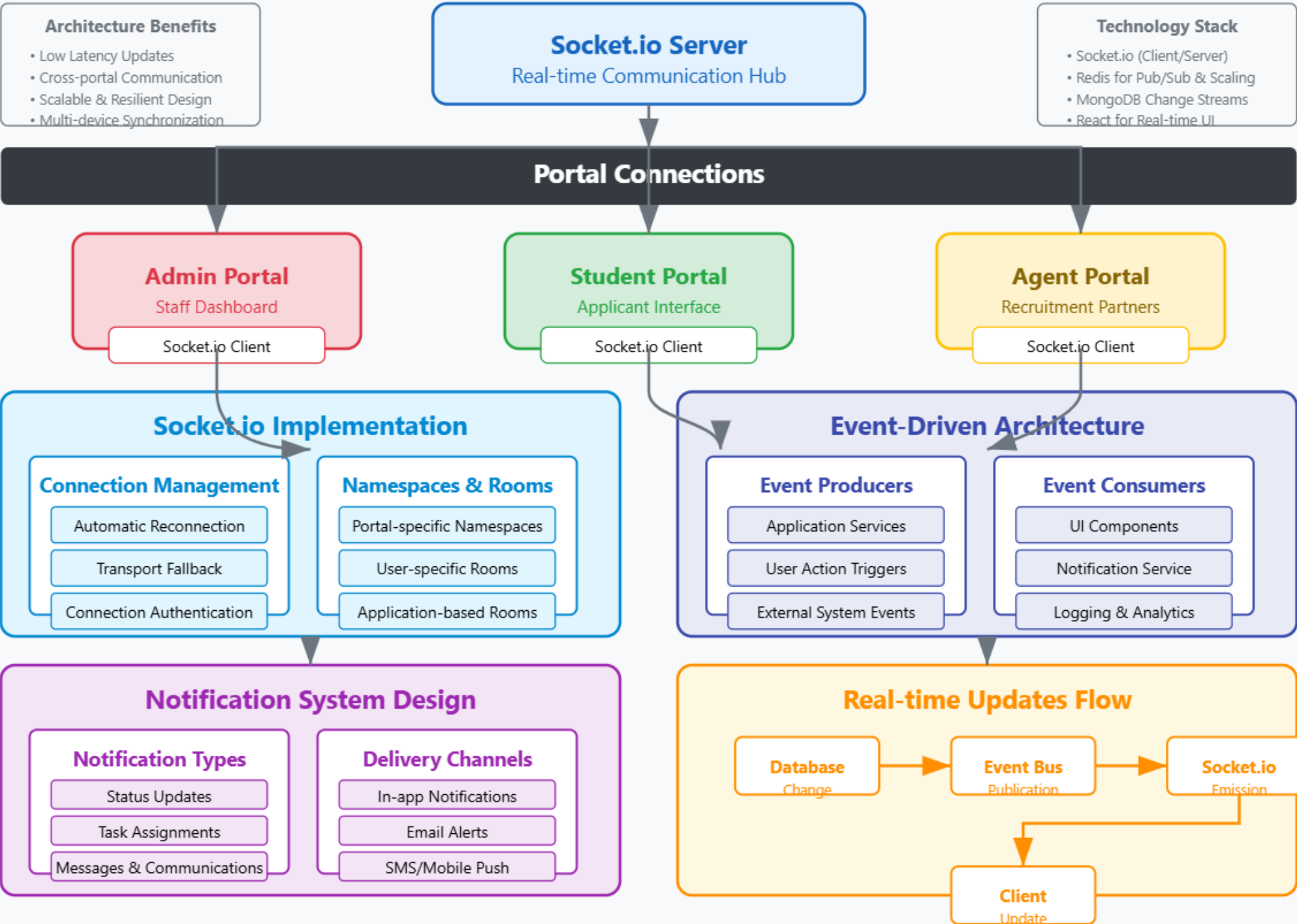
Deployment Strategy



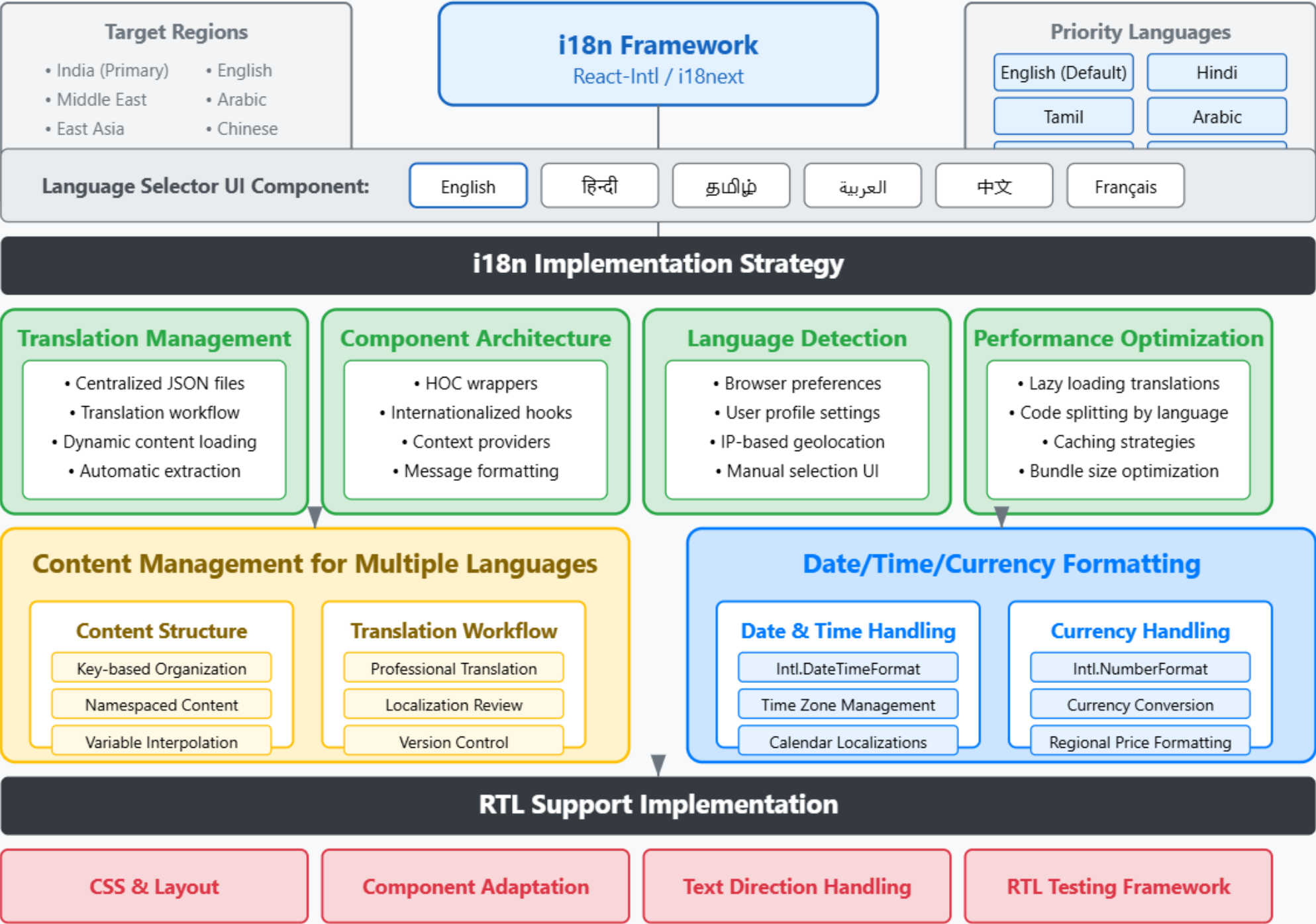
InterEd Admin Portal Microservices Architecture



InterEd Admin Portal Real-time Communication Architecture



InterEd Admin Portal Internationalization Architecture



InterEd Admin Portal User Journey Maps

Critical Admin Workflows

Start Point

End Point

Task Step

Decision Point

Error State

System Response

Flow Path

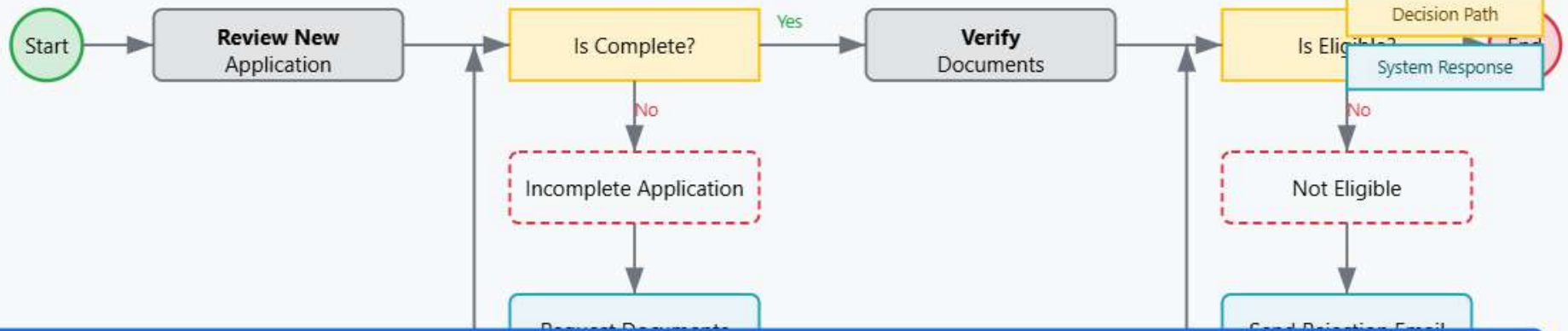
1. Application Processing Workflow

Happy Path

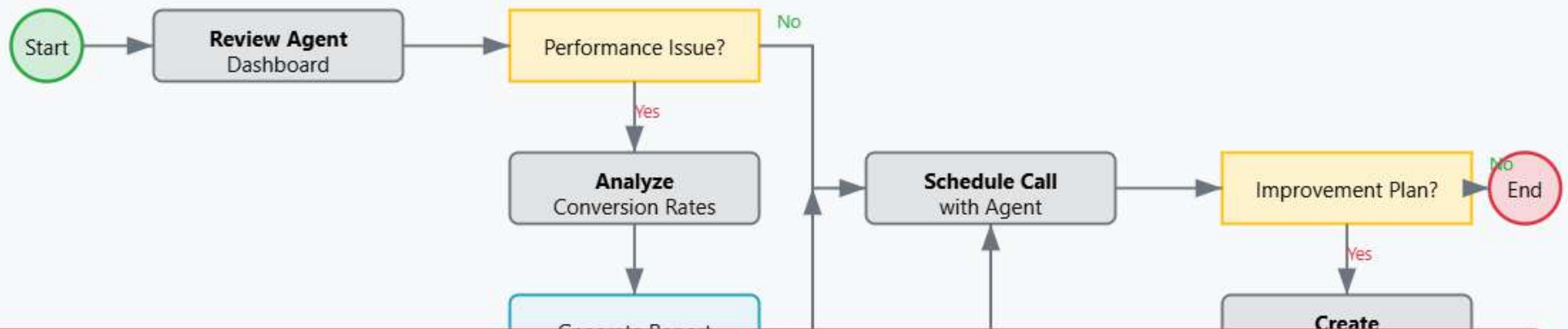
Error Path

Decision Point

System Response



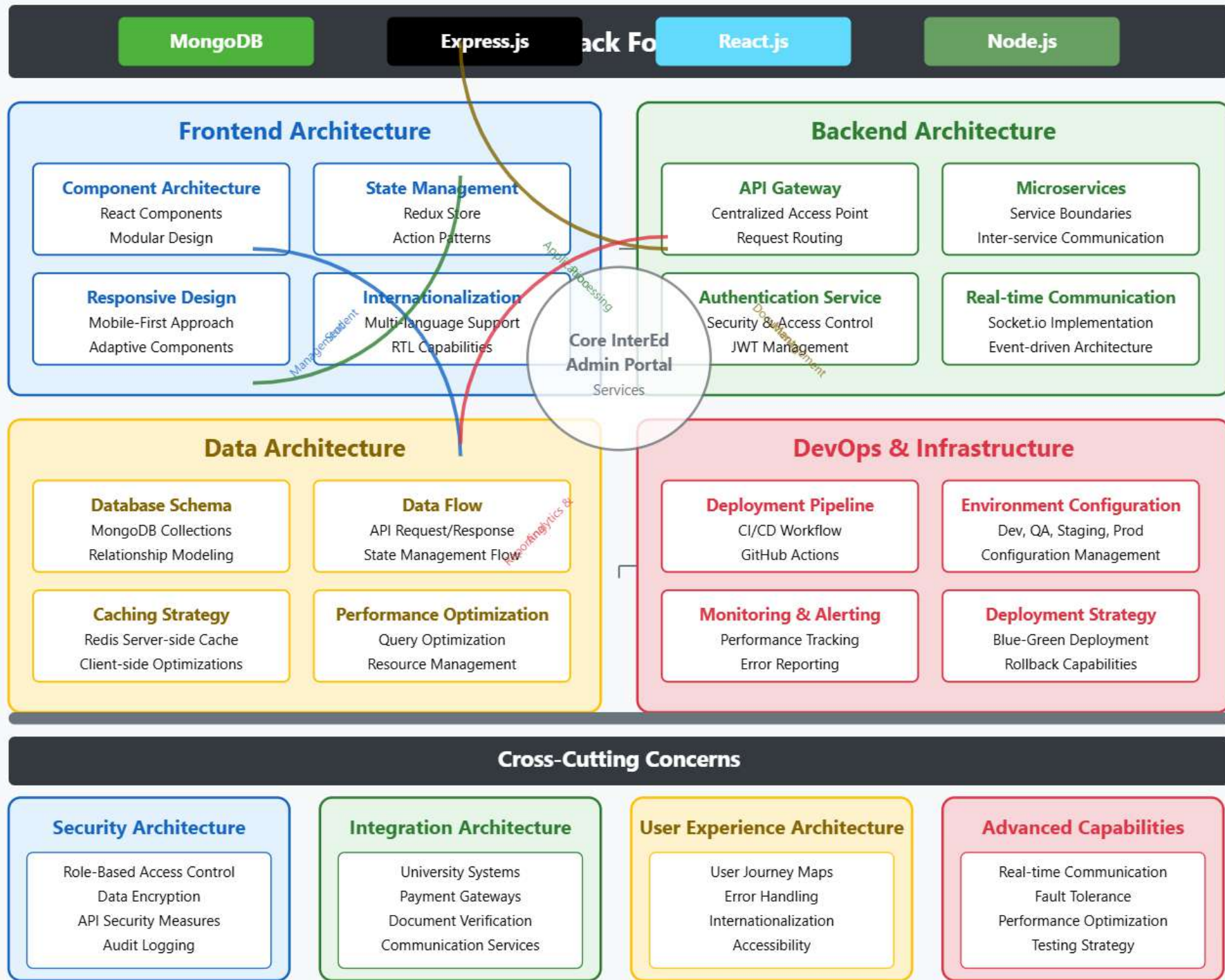
2. Agent Performance Management Workflow



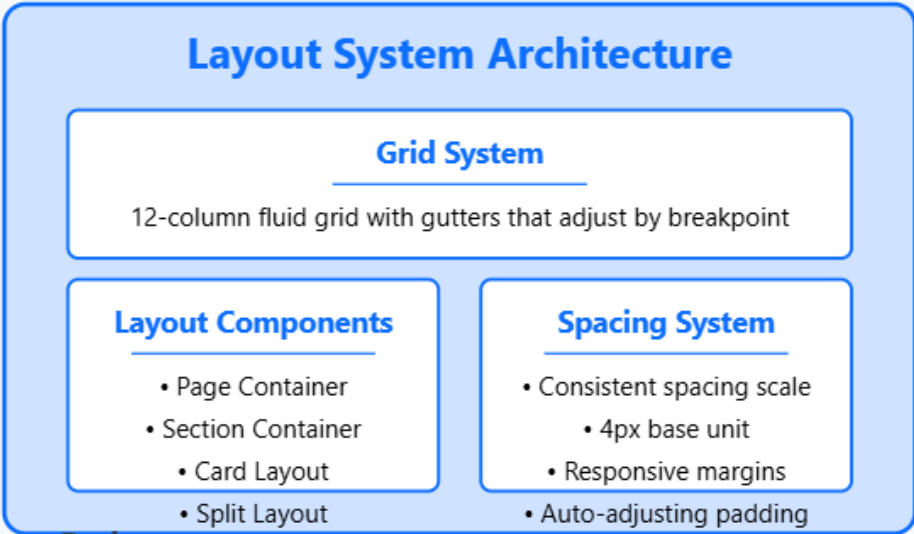
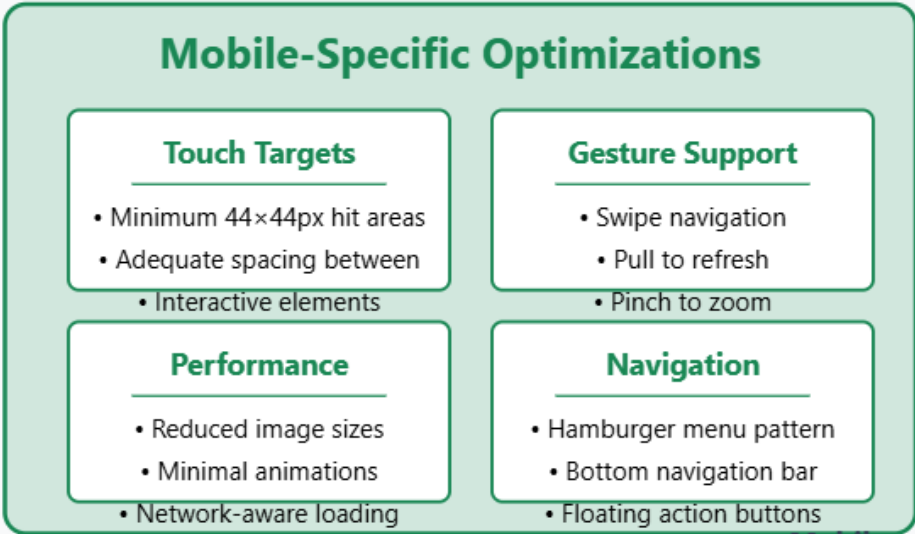
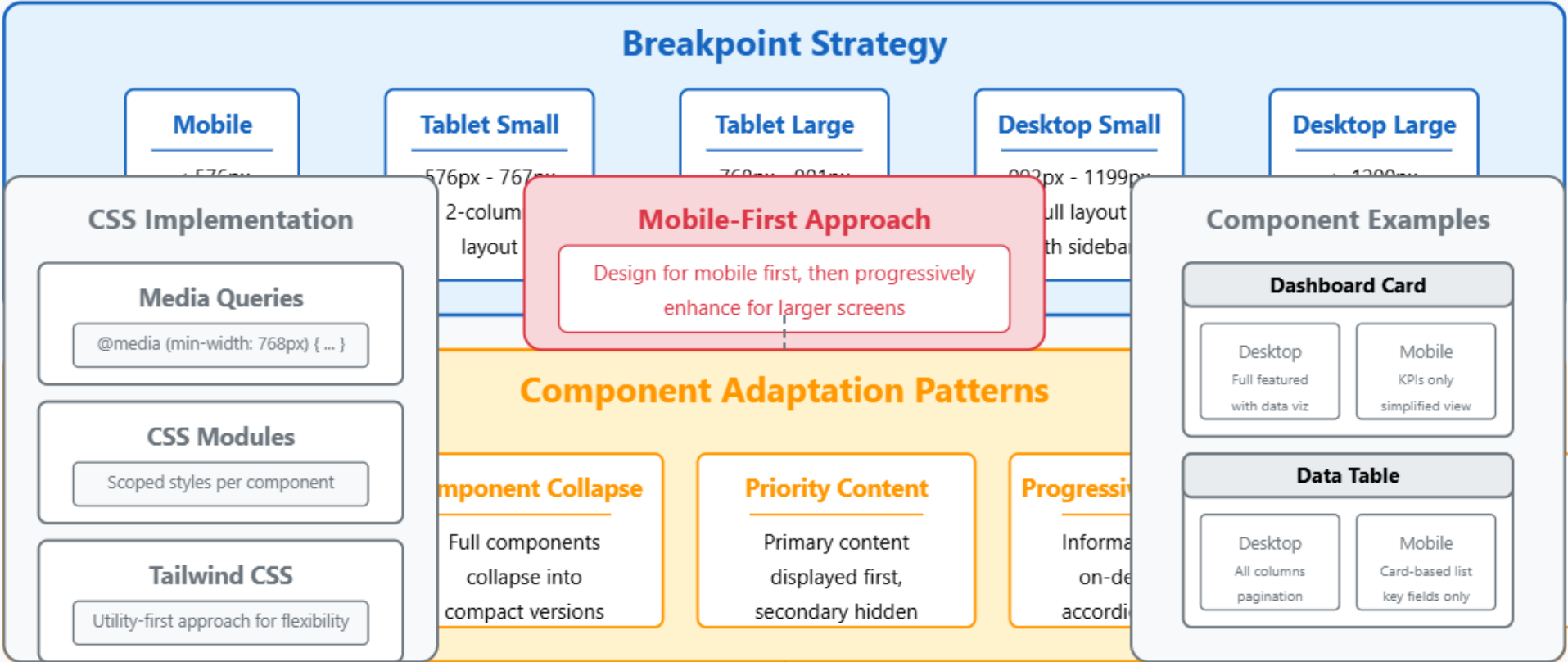
3. Document Verification Error Recovery Path

Additional workflow detailed here

InterEd Admin Portal: Comprehensive Architecture

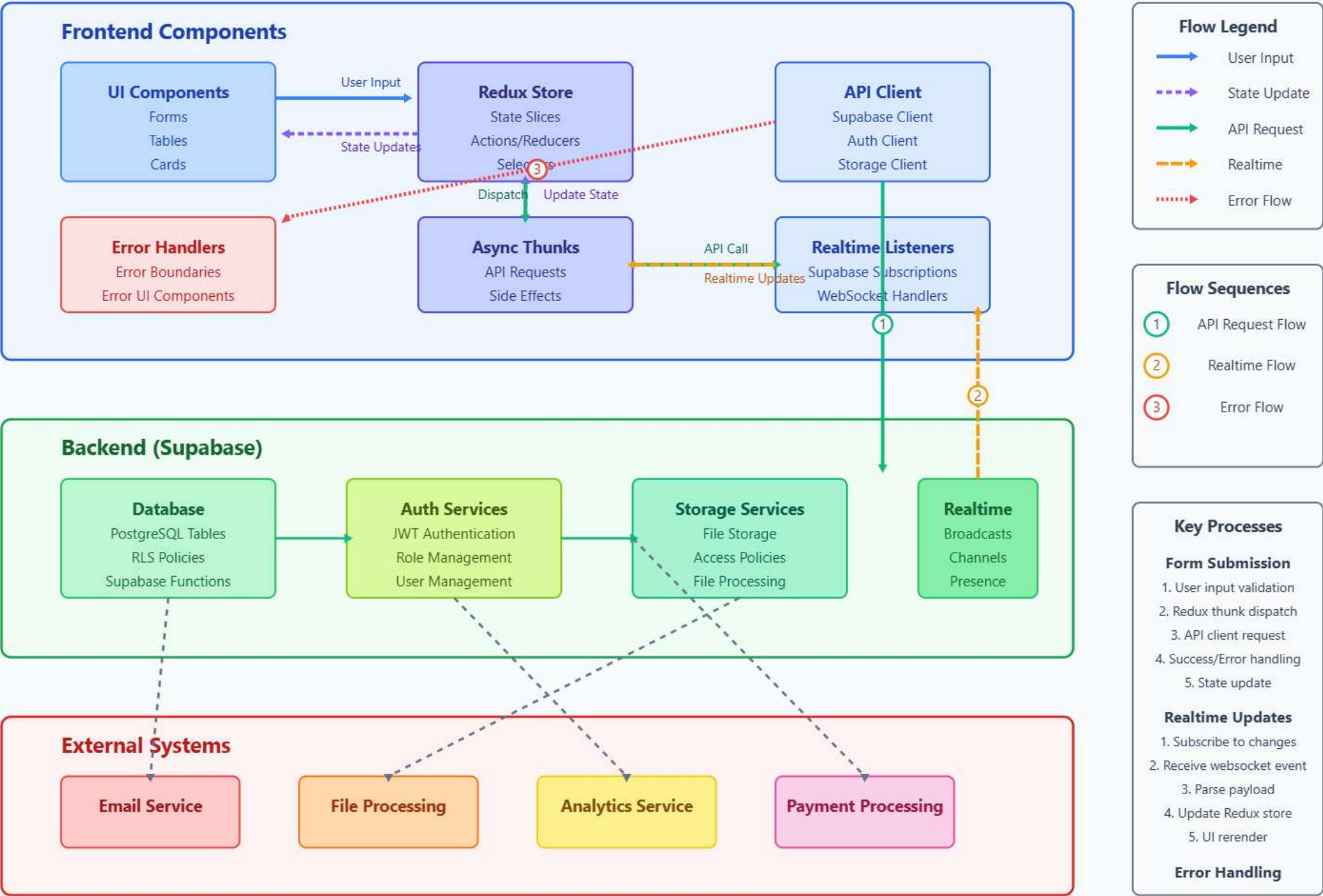


InterEd Admin Portal Responsive Design System

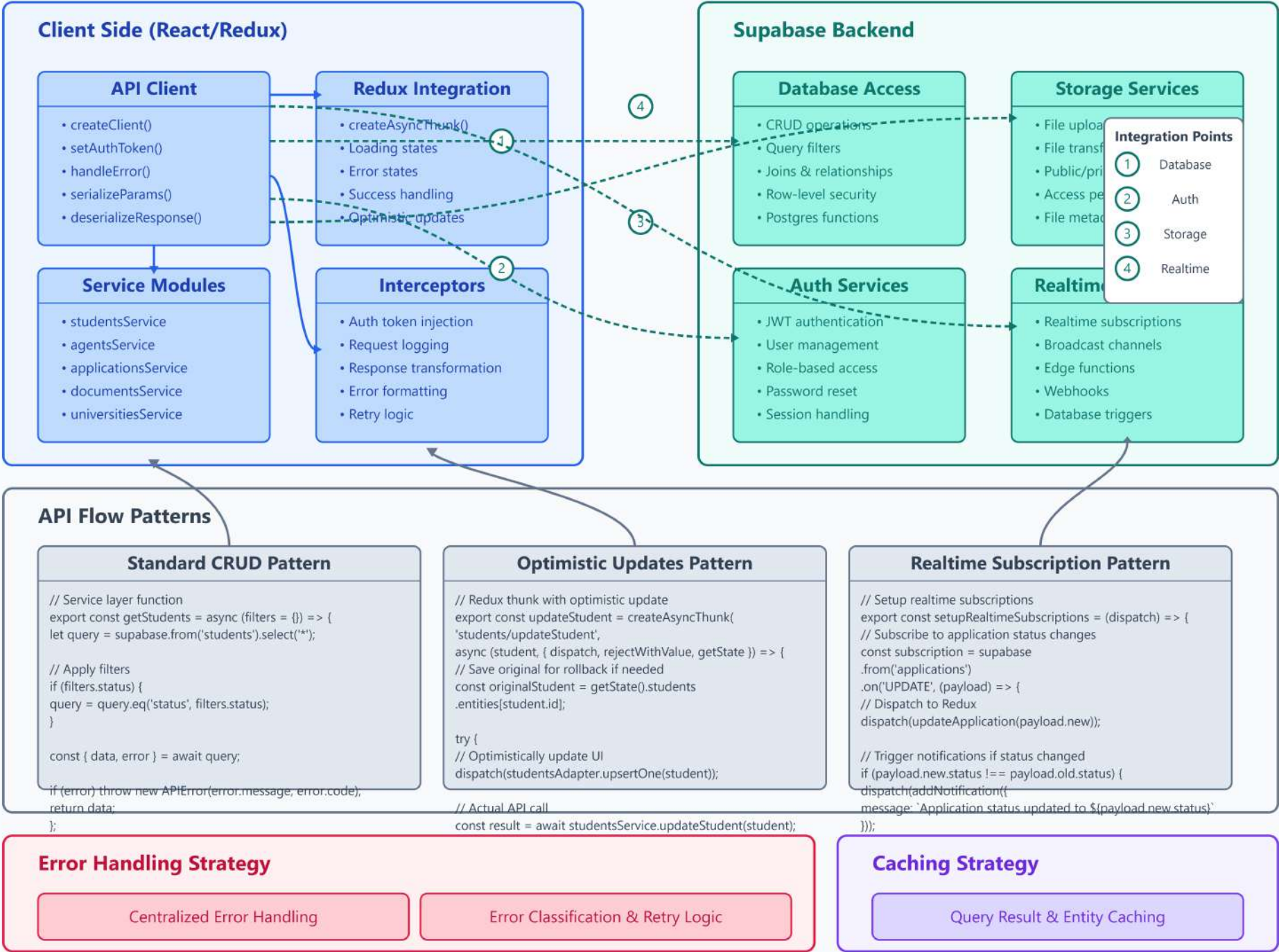


Mobile → Tablet → Desktop

InterEd Admin Portal - Data Flow Architecture

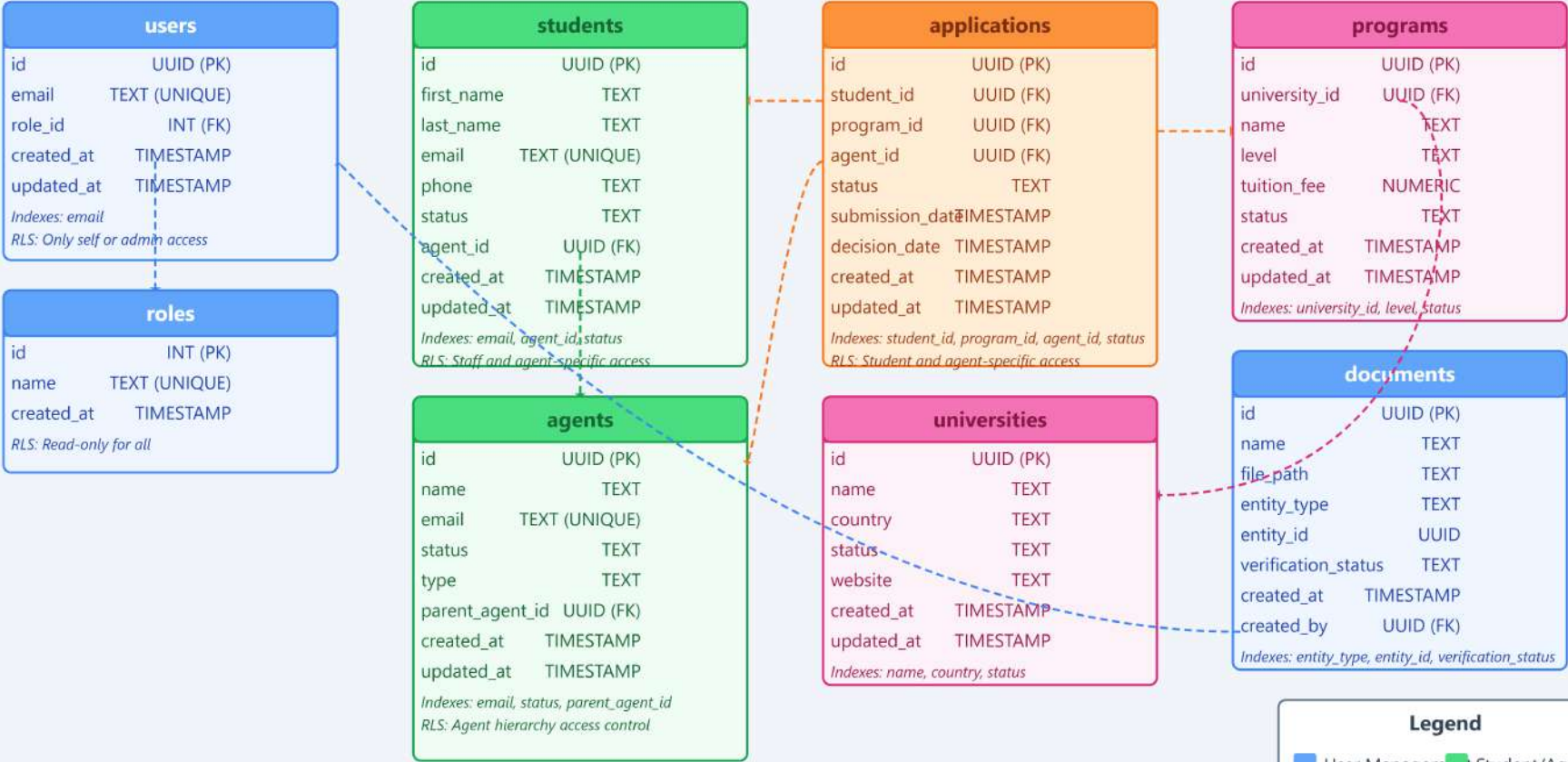


InterEd Admin Portal - API Architecture



InterEd Admin Portal - Database Schema Architecture

Database Schema



Performance Optimization: Indexes on foreign keys, frequently filtered fields (status), and unique fields (email)

Row Level Security Policies

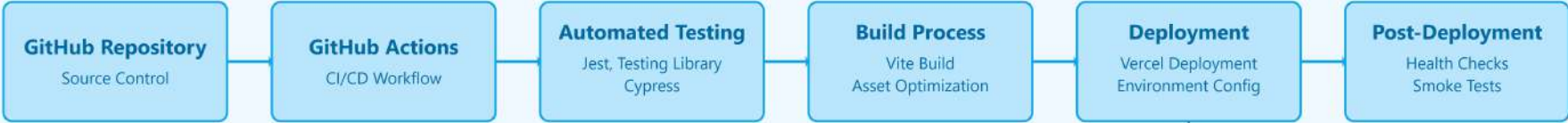
- Users: Only self or admin can view/edit user data
- Students: Staff and associated agent access only
- Applications: Student, agent, and staff access
- Documents: Entity-based access control with ownership policies

Data Validation & Migration Strategy

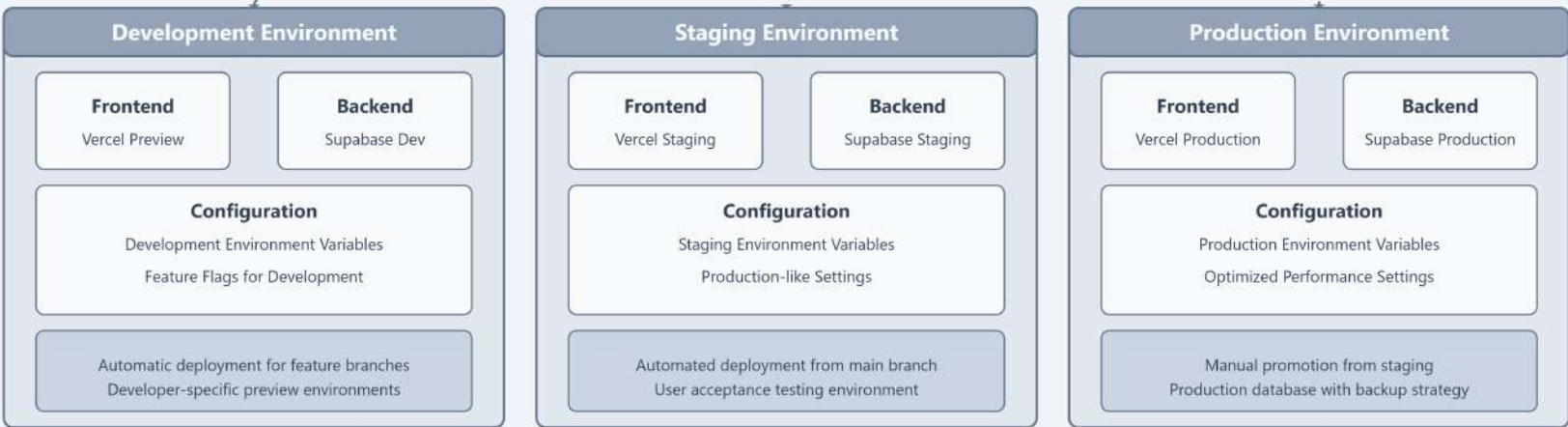
- | Validation Rules | Migration Strategy |
|--|--|
| <ul style="list-style-type: none">Email format validationStatus field enumsRequired field constraintsForeign key constraintsNumeric range validationTimestamp validations | <ul style="list-style-type: none">Version-controlled migrationsUp/down migration scriptsSequential numberingTransaction-based changesSeed data managementCI/CD pipeline integration |

InterEd Admin Portal - Deployment Architecture

CI/CD Pipeline



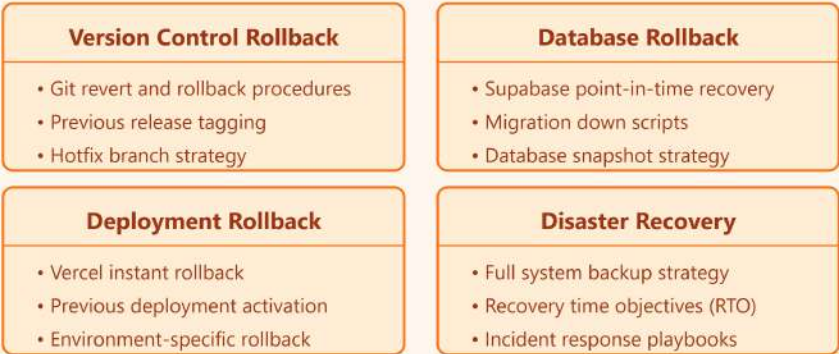
Deployment Environments



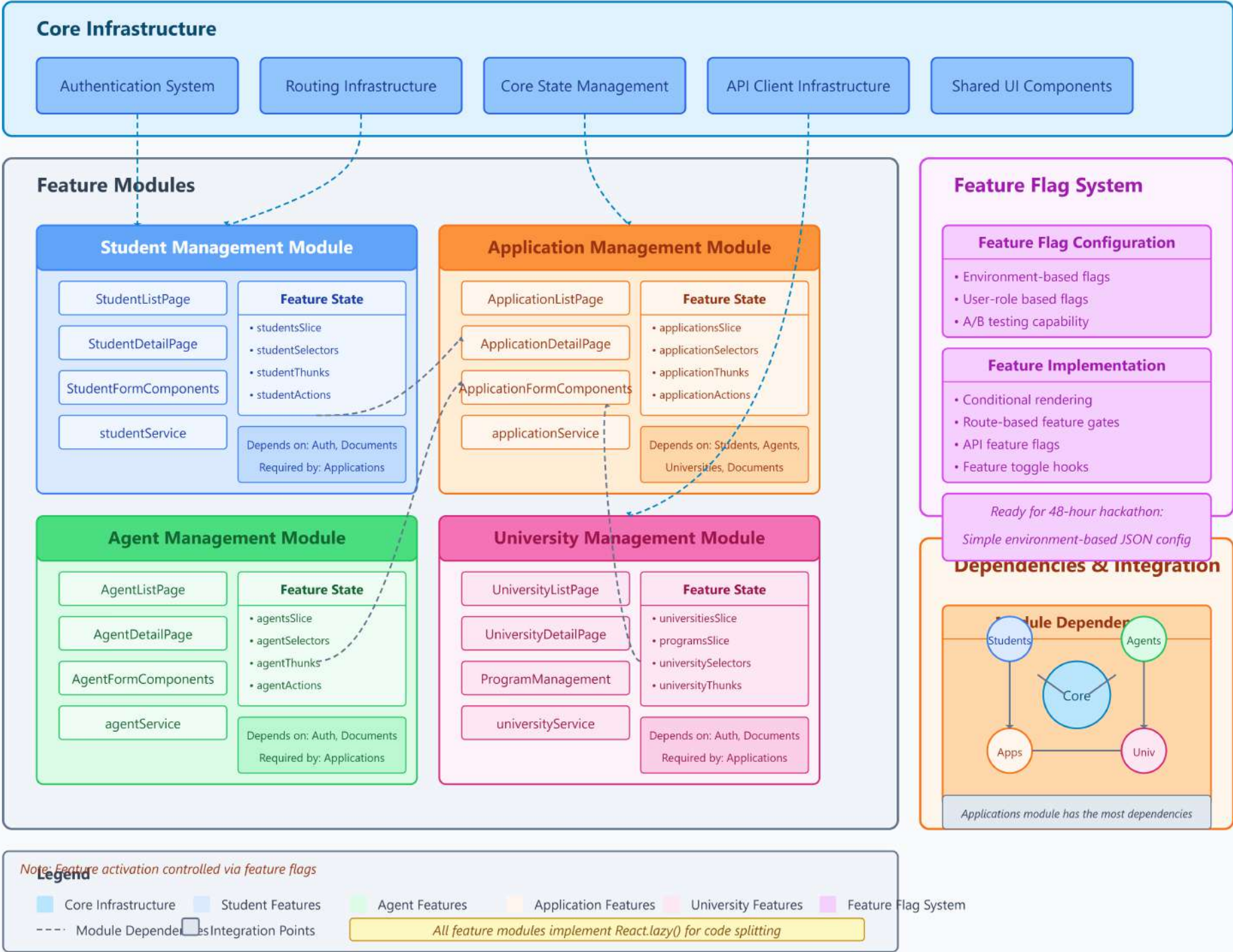
Monitoring & Logging



Rollback & Recovery Strategy



InterEd Admin Portal - Feature Module Architecture



InterEd Admin Portal - Error Handling Architecture



InterEd Admin Portal - Testing Architecture

Test Types & Scope

Unit Tests

- Individual components, functions, hooks
- Redux slices, reducers, selectors Jest
- Utility functions React Testing Library
- Service modules 75-85% Coverage Goal

Integration Tests

- Feature module integration
- Form submissions with validation Jest
- API service integration React Testing Library
- Redux store with 50-60% Coverage Goal

End-to-End Tests

- Critical user flows
- Authentication flows Cypress
- CRUD operations Real Supabase instance
- Multi-step critical paths only (10-15 tests)

Implement unit tests first

Test File Organization

Co-location Strategy

- Unit tests co-located with source files
- Component.jsx and Component.test.jsx
- Makes tests discoverable and maintainable

Tests Directory Structure

- /src/components/ComponentName/ComponentName.test.jsx
- /src/utils/utilName.test.js
- /cypress/integration/features/featureName.spec.js

Consistent naming: Component.test.jsx
Integration tests: feature-name.spec.js

Coverage Targets

- Critical paths: 90%+ coverage
- Core components: 80%+ coverage

Focus Areas

- Auth flows and user permissions
- Data transformation and business logic

48-Hour Hackathon Testing Focus

Focus on unit tests for critical components and utility functions
Prioritize tests for core business logic and data transformation

Mock Strategies

Mock components

Component Mocking

- Mock child components when testing parents
- Use Jest's manual mocks for complex components
- Focus on component interface, not implementation

Service Mocking

- MSW (Mock Service Worker) for API mocking
- Mock Supabase client in tests
- Response fixtures for different scenarios
- Test success, error, and loading states

Generate test data

Test Data Management

Test Fixtures

- JSON fixtures for entity data (students, agents)
- Mock API responses in consistent format E2E tests use custom helpers
- Shared across test suites for consistency

Factory Functions

- Functions to generate test data dynamically
- Customizable entity factories (createStudent)
- Reduces test data maintenance
- Ensures valid entity relationships

Testing Utilities & Helpers

Measure coverage

Custom Render

- Extends React Testing Library's render
- Includes Redux store, Router, Theme providers

Testing Hooks

- Custom renderHook with providers
- Test Redux hooks with store context

User Interaction Helpers

- Form filling helpers (fillForm, submitForm)
- Table interaction helpers (sortTable, filterTable)