# **Sanctuari Platform - Technical Specification v2**

## **Project Overview**

**Product**: Sanctuari - AI-powered insurance procurement and policy administration platform  
 **Market**: B2B businesses in India  
 **Core Value**: Clean, minimalist UI with jargon-free insurance guidance  
 **Design Philosophy**: Trust through clarity, sophistication through simplicity

## **Infrastructure & Domains**

* **Marketing Site**: www.sanctuari.io (Webflow)
* **Application**: platform.sanctuari.io
* **Admin Panel**: admin.sanctuari.io
* **Email System**: bids@sanctuari.io (for all bidder communications)

## **Tech Stack (Defined)**

* **Frontend**: React, Next.js, Node.js, Vanilla CSS (NO Tailwind or CSS libraries)
* **Database**: Supabase
* **Document Storage**: UploadThing
* **Authentication**: Supabase Auth
* **Document Parser**: Llama Parse
* **LLM Architecture**:
  + Claude Sonnet 4 for sub-agents
  + Claude Opus 4.1 for main orchestration agent
* **Agentic Framework**: Langchain
* **Deployment**: Vercel
* **Payments**: Razorpay
* **Email Service**: Brevo

## **UI Design Direction**

### **Brand Colors**

* **Fog**: #F5F4F5 (Light grey background)
* **Iris**: #6F4FFF (Primary purple accent)
* **Rose**: #FD5478 (Alert/highlight pink)
* **Sun**: #F6C754 (Warning/secondary yellow)
* **Ink**: #070921 (Dark navy text/backgrounds)

### **Design Principles**

* **NO EMOJIS** - Maintain professional, clean interface throughout
* **Card-based layouts** with subtle shadows and rounded corners
* **Clean data visualization** using brand colors
* **Generous white space** for breathing room
* **Typography hierarchy**: Clear distinction between headings, body, and captions
* **Subtle animations**: Smooth transitions, no jarring movements
* **Progress indicators**: Linear bars with brand color gradients
* **Interactive elements**: Hover states with gentle color shifts

### **Component Guidelines**

* **Cards**: White background, 8-12px border radius, subtle box-shadow
* **Buttons**: Solid fill for primary (Iris), outlined for secondary, rounded corners
* **Forms**: Clean labels above fields, generous padding, clear focus states
* **Navigation**: Sidebar with icon + text, active state with Iris accent
* **Charts**: Minimal gridlines, brand colors for data points, clean axes
* **Tables**: Alternating row backgrounds (Fog), clear headers, adequate padding
* **Modals**: Centered, overlay with Ink at 40% opacity, smooth entrance

### **Mobile Responsiveness**

* Stack cards vertically on mobile
* Collapsible sidebar navigation
* Touch-friendly tap targets (min 44px)
* Swipeable components where appropriate
* Bottom sheet patterns for mobile actions

## **Core Modules**

### **1. RFQ Creation Module**

#### **Pricing Model**

* First bid: Free
* Subsequent bids: ₹1,599 per placement

#### **UI Structure**

**Layout**: Split-screen interface with Fog background

* **Left Panel** (60% width):
  + Multi-step form with progress bar at top
  + Card-based sections for each step
  + Clean form inputs with Ink text on white backgrounds
* **Right Panel** (40% width):
  + Dynamic guidance panel with Iris accent header
  + Contextual help that updates based on active field
  + Support for text, videos, and images
  + Scrollable content area

#### **CSV Template Processing with Claude API**

**Automated RFQ Structure Generation**:

// Batch processing strategy for CSV templates

const processRFQTemplates = async (csvFiles) => {

// Step 1: Parse CSV files

const templates = await parseCSVBatch(csvFiles);

// Step 2: Batch API calls to Claude Sonnet

const batchSize = 10; // Process 10 templates simultaneously

const batches = chunkArray(templates, batchSize);

// Step 3: Process each batch with rate limiting

const processedTemplates = [];

for (const batch of batches) {

const promises = batch.map(template =>

claudeAPI.process({

model: 'claude-3-sonnet',

task: 'validate\_and\_enhance',

data: {

template,

requirements: [

'Validate input field structure',

'Generate guidance text for each field',

'Suggest field types and validation rules',

'Create jargon-free descriptions'

]

}

})

);

const results = await Promise.all(promises);

processedTemplates.push(...results);

// Rate limiting: 100ms delay between batches

await delay(100);

}

return processedTemplates;

};

**Optimization Strategies**:

* Use Claude Sonnet for cost-effective processing
* Implement caching for repeated field types
* Create template library for common patterns
* Batch similar fields for consistent processing

#### **Flow**

1. **Product Introduction Screen**
   * Clean card layout with Iris accent
   * Icon representation of insurance product
   * 3-4 bullet points of key information
   * "Begin" button with Iris background
2. **Multi-Step Form**
   * Progress bar with completed steps in Iris
   * Field validation with inline error messages in Rose
   * Auto-save with subtle "Saved" indicator
   * Contextual tooltips on hover
3. **RFQ Generation**
   * Processing animation with Iris spinner
   * Success state with checkmark animation
   * Download PDF button with Sun accent for visibility

### **2. Bid Distribution Module**

#### **Distribution Interface Design**

**Two-tier card layout**:

**Tier 1 Card - Direct Contacts**:

* White card with "Your Network" header
* Email input with chip display for added contacts
* Bulk import button with upload icon
* Contact counter in top-right

**Tier 2 Card - Sanctuari Network**:

* Grid view of partner cards
* Each card shows: Logo, Name, Rating, Category
* Multi-select with Iris checkbox overlay
* Search and filter bar at top

### **3. Bid Rules & Settings Module**

#### **Interface Design**

* Tabbed interface with Iris active indicator
* Date picker with calendar widget
* Toggle switches for rules (Iris when active)
* Email template editor with rich text formatting
* Preview panel for email workflows

### **4. Enhanced Bid Submission Portal (Bidder Interface)**

#### **Core Features**

* **RFQ View**: Clean document viewer with zoom controls
* **Quote Submission Interface**:
  + Drag-drop area with dashed border
  + File type indicators (CSV/PDF icons)
  + For Brokers: Insurer selection dropdown before upload
  + Multiple file upload with progress bars

#### **Data Extraction Preview & Edit**

**Preview Interface**:

// After Llama Parse extraction

const QuotePreview = {

layout: 'two-column',

leftPanel: {

title: 'Extracted Information',

sections: [

{

name: 'Coverage Details',

fields: ['Premium', 'Sum Insured', 'Deductible'],

editable: true

},

{

name: 'Key Terms',

fields: ['Policy Period', 'Coverage Type', 'Exclusions'],

editable: true

}

]

},

rightPanel: {

title: 'Original Document',

viewer: 'PDF preview with highlighting'

}

};

**Edit Capabilities**:

* Inline editing with pencil icon
* Validation against RFQ requirements
* "Confirm Extraction" button with Iris background
* "Report Issue" link for extraction errors

#### **Policy Wording Upload**

* Separate upload section for policy documents
* AI analysis triggers for T&C evaluation
* Warning flags in Rose for concerning terms
* Summary card of key findings

### **5. Profile Enhancement System**

#### **Automated Profile Enrichment via Claude API**

const enhanceProfiles = async (profiles) => {

const enhancementBatch = {

model: 'claude-3-sonnet',

batchSize: 20,

tasks: profiles.map(profile => ({

type: 'profile\_enhancement',

data: profile,

instructions: [

'Standardize company information',

'Extract key specializations',

'Generate trust indicators',

'Create searchable tags',

'Summarize in jargon-free language'

]

}))

};

// Process with retry logic

const enhanced = await processBatchWithRetry(enhancementBatch, {

maxRetries: 3,

retryDelay: 1000

});

// Cache results for 24 hours

await cacheProfiles(enhanced, { ttl: 86400 });

return enhanced;

};

### **6. Bid Centre (Command Center)**

#### **Dashboard Design**

**Main View** (Grid Layout):

* Active bids as cards with status indicators
* Color-coded by urgency (Sun for urgent, Iris for active)
* Quick stats bar with key metrics
* Search bar with filter chips below

#### **RFQ Management Interface**

**Layout Structure**:

┌─────────────────────────────────────────┐

│ RFQ Title Status Actions │ (Header Bar)

├──────┬──────────────────────────────────┤

│ Nav │ Main Content Area │

│ │ ┌────────────────────────────┐ │

│ Tabs │ │ Quote Comparison Table │ │

│ │ │ with side-by-side view │ │

│ │ └────────────────────────────┘ │

│ │ ┌────────────────────────────┐ │

│ │ │ AI Insights Panel │ │

│ │ └────────────────────────────┘ │

└──────┴──────────────────────────────────┘

**Navigation Tabs** (Left Sidebar):

1. Overview (dashboard icon)
2. Quotes (document icon)
3. Messages (chat icon)
4. Analytics (chart icon)
5. Settings (gear icon)

**Quote Comparison Engine**:

* Sticky header row with insurer names
* Frozen first column with comparison criteria
* Difference highlighting in Sun yellow
* Best value indicators with Iris badges
* Export button with dropdown (PDF/Excel)

### **7. AI Analysis Architecture**

#### **Multi-Agent System Design**

const QuoteAnalysisSystem = {

mainAgent: {

model: 'claude-opus-4.1',

role: 'orchestrator',

responsibilities: [

'Coordinate sub-agents',

'Synthesize findings',

'Generate final recommendations'

]

},

subAgents: [

{

model: 'claude-sonnet-4',

role: 'coverage\_analyzer',

tasks: ['Extract coverage details', 'Compare limits']

},

{

model: 'claude-sonnet-4',

role: 'pricing\_analyst',

tasks: ['Premium analysis', 'Cost-benefit evaluation']

},

{

model: 'claude-sonnet-4',

role: 'terms\_reviewer',

tasks: ['T&C analysis', 'Flag concerning clauses']

},

{

model: 'claude-sonnet-4',

role: 'compliance\_checker',

tasks: ['Regulatory compliance', 'Document completeness']

}

]

};

## **Claude API Optimization Strategies**

### **1. Batch Processing Architecture**

const BatchProcessor = {

// Queue management

queue: new PriorityQueue(),

// Rate limiting

rateLimiter: {

requestsPerMinute: 50,

tokensPerMinute: 100000

},

// Batching strategy

strategies: {

similarityGrouping: true, // Group similar requests

priorityBatching: true, // Process high-priority first

adaptiveBatchSize: true // Adjust based on response times

},

// Caching

cache: {

ttl: 3600, // 1 hour for guidance text

invalidateOn: ['template\_update', 'profile\_change']

}

};

### **2. Cost Optimization**

* Use Sonnet for routine tasks (80% of requests)
* Reserve Opus 4.1 for complex analysis and final decisions
* Implement prompt templates to reduce token usage
* Cache common responses (field guidance, profile data)

### **3. Performance Metrics**

* Average response time: <2 seconds for Sonnet tasks
* Batch processing: 100 items in <30 seconds
* Cache hit rate target: >60%
* API error rate: <0.1%

## **Admin Panel Module (admin.sanctuari.io)**

### **Core Functionality**

#### **1. RFQ Configuration Management**

* **Dynamic Form Builder**
  + Add/edit/remove questions per insurance product
  + Configure field types (text, number, dropdown, file, date)
  + Set validation rules and required fields
  + Define conditional logic between fields
  + Organize questions into sections/steps
  + Version control for RFQ templates
  + A/B testing different question sets

#### **2. Network Management**

* **Insurer Management**
  + Add/remove insurers from platform
  + Configure insurer profiles and capabilities
  + Set product eligibility per insurer
  + Define email addresses for RFQ distribution
  + Upload insurer logos and documents
  + Set commission structures
  + Performance metrics tracking
* **Broker Management**
  + Onboard new brokers with verification
  + Configure broker profiles and specializations
  + Set broker-insurer relationships
  + Define commission splits
  + Email configuration for each broker
  + Suspension/activation controls
  + Performance dashboards

#### **3. Email Configuration System**

* **Template Management**
  + Create email templates per workflow stage
  + Configure sender details (from bids@sanctuari.io)
  + Set CC/BCC rules per partner
  + Dynamic variable insertion
  + Multi-language support
  + Preview and test functionality
* **Distribution Rules**
  + Map insurers/brokers to specific email addresses
  + Set up email routing logic
  + Configure retry mechanisms
  + Bounce handling rules
  + Delivery tracking integration

#### **4. Platform Configuration**

* **System Settings**
  + Payment gateway configuration
  + API rate limits
  + File upload limits
  + Session timeouts
  + Feature flags for gradual rollouts
  + Maintenance mode controls
* **AI Configuration**
  + Claude API key management
  + Model selection per task
  + Token limits and budgets
  + Prompt template management
  + Response caching rules
  + Fallback mechanisms

#### **5. Comprehensive Audit System**

* **Client Activity Logs**
  + Complete RFQ creation history
  + All document submissions with timestamps
  + Communication logs (emails, messages)
  + Quote submission tracking
  + Payment transactions
  + User session history
* **Document Management**
  + View all uploaded documents per client
  + Download in original format
  + AI extraction results
  + Version history
  + Compliance archival
* **Analytics Dashboard**
  + Real-time platform metrics
  + User behavior analytics
  + Conversion funnels
  + API performance monitoring
  + Error tracking and alerts
  + Custom report builder

#### **6. Support Tools**

* **User Management**
  + View all platform users
  + Access user RFQ history
  + Manual password resets
  + Account verification
  + Refund processing
  + Direct messaging to users
* **Content Management**
  + Update guidance text for RFQ fields
  + Manage help articles
  + Configure tooltips
  + Update legal documents
  + Manage notification content

### **Admin Panel Technical Architecture**

const AdminPanelArchitecture = {

authentication: {

type: 'role-based',

provider: 'Supabase Auth',

roles: ['super\_admin', 'admin', 'support', 'viewer'],

mfa: true

},

frontend: {

framework: 'Next.js',

styling: 'Vanilla CSS',

components: 'Shared with main platform',

routing: 'Protected routes with role checks'

},

api: {

structure: 'RESTful + GraphQL for complex queries',

authentication: 'JWT with role claims',

rateLimit: 'Stricter limits for write operations',

logging: 'All admin actions logged'

},

database: {

schema: 'Separate admin schema in Supabase',

audit: 'Trigger-based audit trails',

backup: 'Hourly snapshots'

},

security: {

ipWhitelisting: true,

activityMonitoring: true,

sessionRecording: 'For support actions',

encryption: 'All sensitive data encrypted at rest'

}

};

### **Admin Panel UI Structure**

┌────────────────────────────────────────────────┐

│ Sanctuari Admin User Profile Logout │

├────────┬───────────────────────────────────────┤

│ │ Main Content Area │

│ Nav │ ┌──────────────────────────────────┐ │

│ │ │ Breadcrumb > Navigation > Path │ │

│ Menu │ ├──────────────────────────────────┤ │

│ │ │ Action Bar with Quick Actions │ │

│ │ ├──────────────────────────────────┤ │

│ │ │ │ │

│ │ │ Dynamic Content Based on │ │

│ │ │ Selected Module │ │

│ │ │ │ │

│ │ └──────────────────────────────────┘ │

└────────┴───────────────────────────────────────┘

## **Development Priorities (Claude Code Optimized)**

Let Claude Code determine the optimal development sequence based on:

* Technical dependencies
* Complexity assessment
* Risk mitigation
* Value delivery timeline
* Integration requirements

Suggested evaluation criteria for Claude Code:

1. Core authentication and database setup
2. Essential user flows
3. AI integration points
4. UI component library
5. Admin panel implementation
6. Testing and deployment pipeline

## **Security Requirements**

* End-to-end encryption for document transmission
* Supabase Row Level Security (RLS) policies
* API key rotation every 30 days
* Document access logging
* GDPR and India's Data Protection Bill compliance
* Secure webhook endpoints for Razorpay

## **Performance Targets**

* First Contentful Paint: <1.5s
* Time to Interactive: <3s
* Lighthouse Score: >90
* API response time: p95 <500ms
* Document processing: <10s for 50-page PDF
* Real-time updates via Supabase subscriptions

## **Error Handling & Monitoring**

* Sentry for error tracking
* Custom error boundaries in React
* Graceful degradation for AI features
* User-friendly error messages (no technical jargon)
* Automated error recovery for batch processes

## **Testing Strategy**

* Unit tests for business logic
* Integration tests for API endpoints
* E2E tests for critical user flows
* AI output validation tests
* Performance testing for batch operations