KYB Tool - Feature Specifications Document

Part 2: Web Dashboard, API Specifications, and Compliance Features

---

Document Information

- **Document Type**: Feature Specifications Document
- **Project**: KYB Tool - Enterprise-Grade Know Your Business Platform
- **Version**: 1.0
- **Date**: January 2025
- **Status**: Final Specification
- **Pages**: Part 2 of 4

---

4. Epic 3: Web Dashboard and User Interface

4.1 Epic Overview

Epic Description: Professional, responsive web dashboard that provides comprehensive business management, risk visualization, and operational control for KYB platform users.

Business Value: Reduces manual workload, improves decision-making speed, and provides self-service capabilities that reduce support burden by 60%.

Success Metrics:

- User task completion rate: >90% for core workflows
- Time to complete merchant review: <5 minutes (down from 20 minutes)
- Dashboard load time: <2 seconds initial load, <500ms navigation

- User satisfaction score: >4.5/5.0

4.2 User Stories

Story 3.1: Merchant Management Dashboard

As a risk analyst
I want a centralized dashboard to view and manage all merchant applications
So that I can efficiently process reviews and track merchant status

Acceptance Criteria:

```
gherkin
```

```
Given I am a logged-in user with merchant management permissions
When I access the merchant dashboard
Then I should see a filterable list of all merchants in my organization
And I can search by merchant name, ID, email, or tax ID
And I can filter by status, risk level, date range, and assigned reviewer
And I can sort by any column (name, status, risk score, date created)
And I can select multiple merchants for bulk actions
And pagination should handle large merchant lists efficiently
And real-time updates should reflect status changes from other users

Scenario: Successful merchant search
Given a database with 10,000+ merchants
When I search for "Acme Corp"
Then results should return within 1 second
And matching merchants should be highlighted
And search should work across name, DBA, and description fields
And fuzzy matching should handle minor typos

Scenario: Advanced filtering
Given I want to find high-risk merchants requiring review
When I apply filters for "Risk Level: High" and "Status: Pending Review"
Then only matching merchants should display
And filter count should show number of results
And filters should be saveable for future use
And clear filters option should reset all filters
```

UI/UX Requirements:

```
typescript
```

```
// Merchant List Component Interface
interface MerchantListProps {
  merchants: Merchant[];
  loading: boolean;
  totalCount: number;
  currentPage: number;
  pageSize: number;
  filters: MerchantFilters;
  sortConfig: SortConfig;
  selectedMerchants: string[];
  onSearch: (query: string) => void;
  onFilter: (filters: MerchantFilters) => void;
  onSort: (field: string, direction: 'asc' | 'desc') => void;
  onSelectMerchant: (merchantId: string, selected: boolean) => void;
  onBulkAction: (action: string, merchantIds: string[]) => void;
  onPageChange: (page: number) => void;
}

interface Merchant {
  id: string;
  legalName: string;
  dbaName?: string;
  status: 'pending' | 'approved' | 'rejected' | 'under_review';
  riskScore: number;
  riskLevel: 'Low' | 'Medium' | 'High' | 'Critical';
  createdAt: string;
  lastUpdated: string;
  assignedTo?: string;
  website?: string;
  industry: string;
  hasFlags: boolean;
  reviewDeadline?: string;
}
```

```
interface MerchantFilters {
  status: string[];
  riskLevel: string[];
  dateRange: { start: string; end: string } | null;
  assignedTo: string[];
  hasFlags: boolean | null;
  industry: string[];
  searchQuery: string;
}
```

Dashboard Layout Specification:

```
jsx
```

```
// Main Dashboard Layout
const MerchantDashboard = () => {
 return (
  <DashboardLayout>
   {/* Header with search and quick actions */}
   <DashboardHeader>
    <SearchBar
     placeholder="Search merchants..."
     onSearch={handleSearch}
     suggestions={searchSuggestions}
    />
    <QuickActions>
     <Button variant="primary" onClick={handleNewMerchant}>
      + Add Merchant
     </Button>
     <Button variant="secondary" onClick={handleBulkImport}>
      Bulk Import
     </Button>
     <NotificationBell count={pendingNotifications} />
    </QuickActions>
   </DashboardHeader>

   {/* Metrics Overview Cards */}
   <MetricsGrid>
    <MetricCard
     title="Pending Reviews"
     value={pendingCount}
     trend="+5 from yesterday"
     color="orange"
     onClick={() => applyFilter({ status: ['pending'] })}
    />
    <MetricCard
     title="High Risk Merchants"
   value={highRiskCount}
   trend="-2 from last week"
   color="red"
   onClick={() => applyFilter({ riskLevel: ['High', 'Critical'] })}
  />
  <MetricCard
   title="Processing Time"
   value="4.2 min avg"
   trend="-30% improvement"
   color="green"
  />
  <MetricCard
   title="Approval Rate"
   value="87.5%"
   trend="+2.1% this month"
   color="blue"
  />
 </MetricsGrid>

 {/* Filters and Controls */}
 <FiltersSection>
  <FilterTabs
   active={activeFilter}
   onChange={setActiveFilter}
   tabs={[
    { id: 'all', label: 'All Merchants', count: totalCount },
    { id: 'pending', label: 'Pending Review', count: pendingCount },
    { id: 'flagged', label: 'Flagged', count: flaggedCount },
    { id: 'assigned', label: 'Assigned to Me', count: assignedCount }
   ]}
  />
  <AdvancedFilters>
   <FilterDropdown
```

```
        label="Risk Level"
        options={riskLevelOptions}
        selected={filters.riskLevel}
        onChange={handleRiskLevelFilter}
      />
      <FilterDropdown
        label="Industry"
        options={industryOptions}
        selected={filters.industry}
        onChange={handleIndustryFilter}
      />
      <DateRangeFilter
        value={filters.dateRange}
        onChange={handleDateRangeFilter}
      />
    </AdvancedFilters>
  </FiltersSection>

  {/* Main Content Area */}
  <ContentArea>
    {/* Bulk Actions Bar */}
    {selectedMerchants.length > 0 && (
      <BulkActionsBar>
        <span>{selectedMerchants.length} selected</span>
        <BulkActionButtons>
          <Button onClick={() => handleBulkAction('approve')}>
            Approve Selected
          </Button>
          <Button onClick={() => handleBulkAction('assign')}>
            Assign to User
          </Button>
          <Button onClick={() => handleBulkAction('export')}>
            Export Selected
      </Button>
    </BulkActionButtons>
  </BulkActionsBar>
)}

{/* Merchant List */}
<MerchantTable>
  <TableHeader>
    <SortableColumn field="legalName">Business Name</SortableColumn>
    <SortableColumn field="status">Status</SortableColumn>
    <SortableColumn field="riskScore">Risk Score</SortableColumn>
    <SortableColumn field="createdAt">Date Created</SortableColumn>
    <Column>Assigned To</Column>
    <Column>Actions</Column>
  </TableHeader>

  <TableBody>
    {merchants.map(merchant => (
      <MerchantRow
        key={merchant.id}
        merchant={merchant}
        selected={selectedMerchants.includes(merchant.id)}
        onSelect={handleMerchantSelect}
        onView={() => openMerchantDetails(merchant.id)}
        onQuickApprove={() => handleQuickAction('approve', merchant.id)}
        onAssign={() => handleAssign(merchant.id)}
      />
    ))}
  </TableBody>
</MerchantTable>

{/* Pagination */}
<Pagination
```

```
        currentPage={currentPage}
        totalPages={totalPages}
        pageSize={pageSize}
        totalCount={totalCount}
        onPageChange={handlePageChange}
        onPageSizeChange={handlePageSizeChange}
      />
    </ContentArea>
  </DashboardLayout>
 );
};
```

Story 3.2: Merchant Detail View and Case Management

As a compliance officer
I want to view comprehensive merchant details in a single interface
So that I can make informed decisions about merchant approval

Acceptance Criteria:

```gherkin
```

```
Given I click on a merchant from the list
When the merchant detail view opens
Then I should see all business information in organized sections
And I should see current risk assessment with explanations
And I should see historical risk trends and changes
And I should see all supporting documentation
And I should see audit trail of all actions taken
And I should have action buttons for approve/reject/request more info
And I can add notes and comments that are saved immediately
And I can assign the case to another user
And all changes should be logged in the audit trail

Scenario: Comprehensive merchant review
Given I am reviewing a medium-risk merchant application
When I open the detailed view
Then I see sections for: Business Info, Risk Assessment, Documentation, History, Actions
And each section should load within 1 second
And risk assessment should show detailed factor breakdown
And I can expand/collapse sections for focused review
And action buttons should be prominent and clearly labeled

Scenario: Document review workflow
Given a merchant has uploaded supporting documents
When I view the merchant details
Then I see all documents organized by type
And I can view documents in-browser without download
And I can mark documents as "reviewed" or "requires attention"
And document status should be saved automatically
And I can request additional documents if needed
```

Story 3.3: Real-time Dashboard Updates and Notifications

As a team manager
I want to see real-time updates on merchant status changes
So that I can monitor team productivity and respond to urgent issues

Acceptance Criteria:

```gherkin
Given multiple users are working on merchant reviews
When a merchant status changes (by any user)
Then all connected users should see the update within 5 seconds
And notification badges should update to reflect new counts
And users should receive browser notifications for high-priority events
And changes should be highlighted temporarily to draw attention
And WebSocket connections should handle network interruptions gracefully

Scenario: Real-time collaboration
Given two users viewing the same merchant list
When User A approves a merchant
Then User B should see the status change immediately
And the merchant should move to the appropriate filtered view
And metrics counters should update in real-time

Scenario: Priority notifications
Given a high-risk merchant is flagged by the system
When the flag is created
Then assigned users should receive immediate browser notification
And the merchant should appear prominently in the dashboard
And notification should include quick action options
```

4.3 Dashboard Technical Implementation

State Management Architecture:

typescript

```typescript
// Redux store structure for dashboard state
interface DashboardState {
 merchants: {
  items: Merchant[];
  totalCount: number;
  loading: boolean;
  error: string | null;
  selectedIds: string[];
  lastUpdated: string;
 };

 filters: {
  active: MerchantFilters;
  saved: SavedFilter[];
  suggestions: FilterSuggestion[];
 };

 ui: {
  currentPage: number;
  pageSize: number;
  sortConfig: SortConfig;
  viewMode: 'list' | 'grid' | 'card';
  sidebarOpen: boolean;
 };

 notifications: {
  items: Notification[];
  unreadCount: number;
  settings: NotificationSettings;
 };

 realtime: {
  connected: boolean;
```

```
    lastHeartbeat: string;
    reconnecting: boolean;
  };
}

// Action creators for merchant management
const merchantActions = {
  // Data loading
  loadMerchants: createAsyncThunk(
    'merchants/load',
    async (params: LoadMerchantsParams) => {
      const response = await api.merchants.list(params);
      return response.data;
    }
  ),

  // Real-time updates
  receiveMerchantUpdate: createAction<MerchantUpdate>('merchants/realtime-update'),

  // Selection management
  selectMerchant: createAction<string>('merchants/select'),
  selectAll: createAction<string[]>('merchants/selectAll'),
  clearSelection: createAction('merchants/clearSelection'),

  // Bulk operations
  bulkApprove: createAsyncThunk(
    'merchants/bulkApprove',
    async (merchantIds: string[]) => {
      const response = await api.merchants.bulkApprove(merchantIds);
      return response.data;
    }
  ),
  // Filtering and search
  setFilters: createAction<MerchantFilters>('merchants/setFilters'),
  saveFilter: createAction<SavedFilter>('merchants/saveFilter'),
  quickSearch: createAsyncThunk(
    'merchants/search',
    async (query: string) => {
      const response = await api.merchants.search(query);
      return response.data;
    }
  )
};
```

Real-time Updates with WebSocket:

```
typescript
```

```
// WebSocket service for real-time updates
class DashboardWebSocketService {
 private ws: WebSocket | null = null;
 private reconnectTimer: NodeJS.Timeout | null = null;
 private heartbeatTimer: NodeJS.Timeout | null = null;

 constructor(
  private dispatch: AppDispatch,
  private getState: () => RootState
 ) {}

 connect() {
  const wsUrl = `${process.env.REACT_APP_WS_URL}/dashboard`;
  const token = this.getState().auth.token;

  this.ws = new WebSocket(`${wsUrl}?token=${token}`);

  this.ws.onopen = this.handleOpen.bind(this);
  this.ws.onmessage = this.handleMessage.bind(this);
  this.ws.onclose = this.handleClose.bind(this);
  this.ws.onerror = this.handleError.bind(this);
 }

 private handleMessage(event: MessageEvent) {
  try {
   const message = JSON.parse(event.data);

   switch (message.type) {
    case 'merchant_update':
     this.dispatch(merchantActions.receiveMerchantUpdate(message.payload));
     this.showNotificationIfRelevant(message.payload);
     break;
```

```
      case 'bulk_operation_complete':
        this.dispatch(notificationActions.add({
          type: 'success',
          title: 'Bulk Operation Complete',
          message: `${message.payload.count} merchants processed`,
          duration: 5000
        }));
        break;

      case 'system_alert':
        this.dispatch(notificationActions.add({
          type: 'warning',
          title: 'System Alert',
          message: message.payload.message,
          persistent: true
        }));
        break;

      case 'heartbeat':
        this.dispatch(realtimeActions.heartbeat());
        break;
    }
  } catch (error) {
    console.error('Failed to parse WebSocket message:', error);
  }
}

private showNotificationIfRelevant(update: MerchantUpdate) {
  const state = this.getState();
  const currentFilters = state.dashboard.filters.active;

  // Check if update affects currently viewed merchants
  const isRelevant = this.checkUpdateRelevance(update, currentFilters);
```

```
    if (isRelevant && update.priority === 'high') {
     // Show browser notification for high-priority updates
     if (Notification.permission === 'granted') {
      new Notification(`Merchant ${update.businessName}`, {
       body: `Status changed to ${update.newStatus}`,
       icon: '/favicon.ico',
       tag: update.merchantId
      });
     }

     // Also show in-app notification
     this.dispatch(notificationActions.add({
      type: 'info',
      title: 'Merchant Status Changed',
      message: `${update.businessName} is now ${update.newStatus}`,
      actions: [
       { label: 'View Details', action: 'view_merchant', data: update.merchantId }
      ]
     }));
    }
   }
  }
```

---

5. Epic 4: API Gateway and Developer Experience

5.1 Epic Overview

Epic Description: Comprehensive RESTful API with authentication, rate limiting, comprehensive documentation, and developer tools that enable seamless integration with customer systems.

Business Value: Reduces integration time from 4-6 weeks to <1 week, increases customer adoption by 40%, and reduces support burden through self-service capabilities.

Success Metrics:

- API response time: <2 seconds (95th percentile)
- API uptime: >99.99%
- Integration time: <1 week for 80% of customers
- Documentation satisfaction: >4.5/5.0 stars

5.2 User Stories

Story 4.1: API Authentication and Security

As a backend developer
I want secure and easy-to-use API authentication
So that I can integrate KYB services without compromising security

Acceptance Criteria:

```
gherkin
```

```
Given I am a registered user with API access
When I generate an API key through the dashboard
Then I should receive a secure API key with configurable permissions
And I can use the API key in the Authorization header
And the API key should have rate limiting based on my plan
And I can regenerate or revoke API keys at any time
And API usage should be tracked and displayed in my dashboard
And failed authentication attempts should be logged and monitored

Scenario: Successful API key generation
Given I have API access permissions
When I create a new API key with "read" and "classify" permissions
Then I receive a unique API key starting with "kyb_"
And the key should work immediately for authorized endpoints
And usage tracking should begin for this key
And I can see the key permissions in my dashboard

Scenario: API key security
Given I have an active API key
When I make requests with the key in the header
Then requests should be processed normally
And when I make requests without the key
Then I should receive a 401 Unauthorized response
And when I use a revoked key
Then I should receive a 403 Forbidden response with clear error message
```

API Authentication Implementation:

```yaml
```

```yaml
# API Key Authentication Specification
Authentication:
  method: "API Key"
  header: "Authorization: Bearer {api_key}"
  format: "kyb_{version}_{random_string}"
  example: "kyb_v1_1a2b3c4d5e6f7g8h9i0j"

Rate_Limiting:
  starter_plan:
    requests_per_minute: 100
    requests_per_hour: 5000
    requests_per_day: 50000
  professional_plan:
    requests_per_minute: 500
    requests_per_hour: 25000
    requests_per_day: 500000
  enterprise_plan:
    requests_per_minute: 2000
    requests_per_hour: 100000
    requests_per_day: 2000000

Response_Headers:
  - "X-RateLimit-Limit: 100"
  - "X-RateLimit-Remaining: 87"
  - "X-RateLimit-Reset: 1640995200"
  - "X-Request-ID: req_1234567890abcdef"

Error_Responses:
  401:
    error: "unauthorized"
    message: "Invalid or missing API key"
    code: "INVALID_API_KEY"
  403:
    error: "forbidden"
    message: "API key does not have required permissions"
    code: "INSUFFICIENT_PERMISSIONS"
  429:
    error: "rate_limit_exceeded"
    message: "Rate limit exceeded. Retry after 60 seconds"
    code: "RATE_LIMIT_EXCEEDED"
    retry_after: 60
```

Story 4.2: Core API Endpoints

As a systems integrator
I want well-documented API endpoints for all KYB operations
So that I can build robust integrations with predictable behavior

Acceptance Criteria:

```
gherkin
```

```
Given I have valid API credentials
When I make requests to core API endpoints
Then responses should follow consistent patterns
And error handling should be predictable and informative
And response times should be under 2 seconds
And responses should include request IDs for troubleshooting
And API versioning should be clearly indicated
And backward compatibility should be maintained for major versions

Scenario: Business classification API
Given valid business data
When I POST to /api/v1/classify
Then I receive classification results within 2 seconds
And response includes confidence scores
And response follows documented schema exactly
And alternative suggestions are included when appropriate

Scenario: Batch operations
Given a batch of business records
When I POST to /api/v1/classify/batch
Then I receive a job ID immediately
And I can track progress via GET /api/v1/jobs/{job_id}
And webhook notifications are sent when configured
And results are available for 7 days after completion
```

Core API Specification:

```yaml
```

```yaml
# Core API Endpoints
base_url: "https://api.kybtool.com"
version: "v1"

endpoints:
  # Business Classification
  classify_business:
    path: "/api/v1/classify"
    method: "POST"
    description: "Classify a single business"
    rate_limit: "Standard plan limits apply"
    request_schema:
      type: "object"
      required: ["business_description"]
      properties:
        business_description:
          type: "string"
          minLength: 10
          maxLength: 1000
        business_name:
          type: "string"
          maxLength: 200
        website_url:
          type: "string"
          format: "uri"
        country:
          type: "string"
          pattern: "^[A-Z]{2}$"
          default: "US"
        include_similar:
          type: "boolean"
          default: false
    response_schema:
```

```yaml
    type: "object"
    properties:
      classification_id:
        type: "string"
      primary_classifications:
        type: "object"
        properties:
          mcc: {code, description, confidence}
          naics: {code, description, confidence}
          sic: {code, description, confidence}
      processing_time_ms:
        type: "integer"
      timestamp:
        type: "string"
        format: "date-time"

# Batch Classification
classify_batch:
  path: "/api/v1/classify/batch"
  method: "POST"
  description: "Classify multiple businesses"
  max_batch_size: 1000
  request_schema:
    type: "object"
    required: ["businesses"]
    properties:
      businesses:
        type: "array"
        maxItems: 1000
        items:
          type: "object"
          required: ["id", "business_description"]
          properties:
```

```yaml
      id: {type: "string"}
      business_description: {type: "string"}
      # ... other fields same as single classification
    webhook_url:
     type: "string"
     format: "uri"
    notification_email:
     type: "string"
     format: "email"
  response_schema:
   type: "object"
   properties:
    job_id: {type: "string"}
    status: {enum: ["queued", "processing", "completed", "failed"]}
    estimated_completion: {type: "string", format: "date-time"}

# Risk Assessment
assess_risk:
  path: "/api/v1/risk/assess"
  method: "POST"
  description: "Perform risk assessment on business"
  request_schema:
   type: "object"
   required: ["business_id"]
   properties:
    business_id: {type: "string"}
    assessment_type:
     enum: ["initial", "periodic", "triggered"]
     default: "initial"
    include_predictions:
     type: "boolean"
     default: false
    risk_tolerance:
        enum: ["conservative", "moderate", "aggressive"]
        default: "moderate"
  response_schema:
    type: "object"
    properties:
      assessment_id: {type: "string"}
      overall_score: {type: "integer", minimum: 1, maximum: 100}
      risk_level: {enum: ["Low", "Medium", "High", "Critical"]}
      risk_categories: {type: "object"}
      recommendations: {type: "array"}
      assessed_at: {type: "string", format: "date-time"}

# Business Management
create_business:
  path: "/api/v1/businesses"
  method: "POST"
  description: "Create new business profile"

get_business:
  path: "/api/v1/businesses/{business_id}"
  method: "GET"
  description: "Retrieve business profile"

update_business:
  path: "/api/v1/businesses/{business_id}"
  method: "PATCH"
  description: "Update business profile"

list_businesses:
  path: "/api/v1/businesses"
  method: "GET"
  description: "List businesses with filtering and pagination"
  parameters:
    - name: "page"
      type: "integer"
      default: 1
    - name: "limit"
      type: "integer"
      default: 50
      maximum: 1000
    - name: "status"
      type: "string"
      enum: ["active", "inactive", "pending", "suspended"]
    - name: "risk_level"
      type: "string"
      enum: ["Low", "Medium", "High", "Critical"]
    - name: "created_after"
      type: "string"
      format: "date-time"

# Webhook Management
list_webhooks:
  path: "/api/v1/webhooks"
  method: "GET"
  description: "List configured webhooks"

create_webhook:
  path: "/api/v1/webhooks"
  method: "POST"
  description: "Create new webhook endpoint"
  request_schema:
    type: "object"
    required: ["url", "events"]
    properties:
      url:
        type: "string"
        format: "uri"
      events:
        type: "array"
        items:
          enum: ["classification.completed", "risk.assessed", "business.created"]
      secret:
        type: "string"
        minLength: 32
      active:
        type: "boolean"
        default: true
```

Story 4.3: Interactive API Documentation

As a developer evaluating the KYB API
I want comprehensive, interactive documentation
So that I can understand capabilities and test integration quickly

Acceptance Criteria:

```
gherkin
```

```gherkin
Given I visit the API documentation site
When I browse the documentation
Then I should see all endpoints with detailed descriptions
And I should be able to test API calls directly from the documentation
And I should see example requests and responses for each endpoint
And I should see authentication requirements clearly explained
And I should see rate limits and error codes documented
And I should be able to generate code examples in multiple languages
And documentation should be searchable and well-organized

Scenario: Interactive API testing
Given I am viewing the /classify endpoint documentation
When I enter sample data in the interactive form
And I click "Try it out"
Then I should see the actual API request being made
And I should receive a real response from the API
And the response should be formatted for easy reading
And I should be able to copy the generated code

Scenario: Code generation
Given I am viewing any API endpoint
When I select "Show code examples"
Then I should see examples in Python, JavaScript, Java, and cURL
And examples should include proper authentication
And examples should handle error responses appropriately
And I should be able to copy examples with one click
```

---

6. Epic 5: Compliance and Sanctions Screening

6.1 Epic Overview

Epic Description: Comprehensive compliance screening system that checks businesses and individuals against global sanctions lists, regulatory databases, and adverse media sources.

Business Value: Ensures regulatory compliance, prevents onboarding sanctioned entities, and provides audit-ready documentation for regulatory examinations.

Success Metrics:

- Sanctions screening accuracy: >99.5% (minimal false positives)
- Screening response time: <5 seconds for comprehensive check
- Regulatory coverage: OFAC, UN, EU, UK sanctions lists
- Audit trail completeness: 100% of screening decisions logged

6.2 User Stories

Story 5.1: Automated Sanctions Screening

As a compliance manager
I want automated screening against global sanctions lists
So that I can prevent onboarding sanctioned entities and maintain compliance

Acceptance Criteria:

```
gherkin
```

```
Given a new business application
When sanctions screening is performed
Then the system should check against all relevant sanctions lists
And screening should complete within 5 seconds
And any matches should be flagged with severity levels
And partial/fuzzy matches should be scored for review
And screening results should be permanently stored
And audit logs should record all screening decisions
And false positives should be easy to clear and document

Scenario: Clean screening result
Given a business "ABC Manufacturing Ltd" with no sanctions matches
When sanctions screening is performed
Then screening should return "clear" status
And all checked lists should be documented
And screening timestamp and version should be recorded
And result should be cached for future reference

Scenario: Potential sanctions match
Given a business name similar to a sanctioned entity
When screening detects a partial match
Then match should be flagged for manual review
And similarity score should be provided
And reviewer should see side-by-side comparison
And reviewer can mark as false positive with reasoning
And decision should update the business profile immediately
```

Sanctions Screening Implementation:

```
python
```

```python
# Sanctions screening service specification
class SanctionsScreeningService:
    """
    Comprehensive sanctions screening against multiple lists
    """

    def __init__(self):
        self.screening_lists = {
            'ofac_sdn': {
                'name': 'OFAC Specially Designated Nationals',
                'update_frequency': 'weekly',
                'priority': 'high',
                'fuzzy_threshold': 0.85
            },
            'ofac_ssi': {
                'name': 'OFAC Sectoral Sanctions Identifications',
                'update_frequency': 'weekly',
                'priority': 'high',
                'fuzzy_threshold': 0.90
            },
            'un_consolidated': {
                'name': 'UN Consolidated Sanctions List',
                'update_frequency': 'daily',
                'priority': 'high',
                'fuzzy_threshold': 0.85
            },
            'eu_sanctions': {
                'name': 'EU Financial Sanctions Database',
                'update_frequency': 'weekly',
                'priority': 'medium',
                'fuzzy_threshold': 0.88
            },
            'uk_sanctions': {
```

```
            'name': 'UK Financial Sanctions',
            'update_frequency': 'weekly',
            'priority': 'medium',
            'fuzzy_threshold': 0.88
        }
    }

    self.screening_algorithms = {
        'exact_match': ExactMatchScreener(),
        'fuzzy_match': FuzzyMatchScreener(),
        'phonetic_match': PhoneticMatchScreener(),
        'alias_match': AliasMatchScreener()
    }

async def screen_entity(self, entity_data: dict) -> ScreeningResult:
    """
    Perform comprehensive sanctions screening
    """
    screening_request = ScreeningRequest(
        entity_name=entity_data.get('legal_name'),
        aliases=entity_data.get('aliases', []),
        addresses=entity_data.get('addresses', []),
        tax_ids=entity_data.get('tax_ids', []),
        country=entity_data.get('country'),
        entity_type='business'
    )

    # Screen against all lists in parallel
    screening_tasks = []
    for list_id, list_config in self.screening_lists.items():
        task = self.screen_against_list(screening_request, list_id, list_config)
        screening_tasks.append(task)
    # Wait for all screenings to complete
    list_results = await asyncio.gather(*screening_tasks)

    # Consolidate results
    consolidated_result = self.consolidate_screening_results(
        list_results, screening_request
    )

    # Store screening record
    await self.store_screening_record(consolidated_result)

    return consolidated_result

async def screen_against_list(self, request: ScreeningRequest,
                              list_id: str, config: dict) -> ListScreeningResult:
    """
    Screen against a specific sanctions list
    """
    matches = []

    # Try different matching algorithms
    for algo_name, algorithm in self.screening_algorithms.items():
        algo_matches = await algorithm.screen(
            request, list_id, config['fuzzy_threshold']
        )
        matches.extend(algo_matches)

    # Deduplicate and score matches
    unique_matches = self.deduplicate_matches(matches)
    scored_matches = self.score_matches(unique_matches, config)

    return ListScreeningResult(
        list_id=list_id,
            list_name=config['name'],
            matches=scored_matches,
            screened_at=datetime.utcnow(),
            algorithm_versions=self.get_algorithm_versions()
        )

# Screening result schemas
@dataclass
class ScreeningMatch:
    entity_name: str
    list_entry_id: str
    match_type: str  # 'exact', 'fuzzy', 'phonetic', 'alias'
    similarity_score: float
    match_details: dict
    severity: str  # 'high', 'medium', 'low'
    requires_review: bool

@dataclass
class ScreeningResult:
    screening_id: str
    entity_data: dict
    overall_status: str  # 'clear', 'match', 'potential_match', 'error'
    risk_score: int  # 1-100
    matches: List[ScreeningMatch]
    lists_checked: List[str]
    screened_at: datetime
    expires_at: datetime
    reviewer_notes: str = None
    cleared_by: str = None
    cleared_at: datetime = None
```

Story 5.2: Adverse Media Monitoring

As a risk officer

I want to monitor businesses for negative news and media mentions

So that I can identify reputational risks and regulatory violations

Acceptance Criteria:

```gherkin
```

```
Given a business profile in the system
When adverse media monitoring is performed
Then the system should search news sources for negative mentions
And search should cover the last 24 months of news
And results should be categorized by risk severity
And duplicate articles should be consolidated
And search should handle company name variations
And results should be scored for relevance and credibility
And monitoring should be performed periodically for active businesses

Scenario: Significant adverse media found
Given a business with recent legal violations
When media monitoring runs
Then relevant news articles should be identified
And articles should be scored for severity
And business risk profile should be updated
And compliance team should be notified
And articles should be stored for audit purposes

Scenario: False positive management
Given a business with a common name
When media monitoring finds irrelevant articles
Then system should score relevance accurately
And low-relevance articles should be filtered out
And manual review tools should be available
And learning from manual reviews should improve future screening
```

This completes Part 2 of the Feature Specifications Document, covering the Web Dashboard, comprehensive API specifications, and Compliance/Sanctions screening features with detailed user stories and technical implementations.

Should I continue with **Part 3: Advanced Features, Analytics, and Integration Requirements**?