# **Sanctuari Component Library & User Stories**

## **Typography System (Geist Font)**

### **Font Implementation**

@font-face {

font-family: 'Geist';

src: url('/fonts/Geist-Variable.woff2') format('woff2-variations');

font-weight: 100 900;

font-display: swap;

}

@font-face {

font-family: 'Geist Mono';

src: url('/fonts/GeistMono-Variable.woff2') format('woff2-variations');

font-weight: 100 900;

font-display: swap;

}

### **Type Scale**

/\* Headings \*/

.heading-1 {

font-family: 'Geist', sans-serif;

font-size: 48px;

font-weight: 600;

line-height: 1.2;

letter-spacing: -0.02em;

color: var(--ink);

}

.heading-2 {

font-family: 'Geist', sans-serif;

font-size: 36px;

font-weight: 600;

line-height: 1.3;

letter-spacing: -0.01em;

color: var(--ink);

}

.heading-3 {

font-family: 'Geist', sans-serif;

font-size: 28px;

font-weight: 500;

line-height: 1.4;

color: var(--ink);

}

.heading-4 {

font-family: 'Geist', sans-serif;

font-size: 20px;

font-weight: 500;

line-height: 1.4;

color: var(--ink);

}

/\* Body Text \*/

.body-large {

font-family: 'Geist', sans-serif;

font-size: 18px;

font-weight: 400;

line-height: 1.6;

color: var(--ink);

}

.body-regular {

font-family: 'Geist', sans-serif;

font-size: 16px;

font-weight: 400;

line-height: 1.5;

color: var(--ink);

}

.body-small {

font-family: 'Geist', sans-serif;

font-size: 14px;

font-weight: 400;

line-height: 1.5;

color: var(--ink);

}

/\* Supporting Text \*/

.caption {

font-family: 'Geist', sans-serif;

font-size: 12px;

font-weight: 400;

line-height: 1.4;

color: var(--ink);

opacity: 0.7;

}

.label {

font-family: 'Geist', sans-serif;

font-size: 14px;

font-weight: 500;

line-height: 1;

text-transform: uppercase;

letter-spacing: 0.05em;

color: var(--ink);

}

/\* Monospace (for data/numbers) \*/

.mono {

font-family: 'Geist Mono', monospace;

font-size: 14px;

font-weight: 400;

color: var(--ink);

}

## **Component Library**

### **1. Color System Variables**

:root {

--fog: #F5F4F5;

--iris: #6F4FFF;

--rose: #FD5478;

--sun: #F6C754;

--ink: #070921;

--white: #FFFFFF;

/\* Opacity variants \*/

--ink-80: rgba(7, 9, 33, 0.8);

--ink-60: rgba(7, 9, 33, 0.6);

--ink-40: rgba(7, 9, 33, 0.4);

--ink-20: rgba(7, 9, 33, 0.2);

--ink-10: rgba(7, 9, 33, 0.1);

--iris-light: rgba(111, 79, 255, 0.1);

--rose-light: rgba(253, 84, 120, 0.1);

--sun-light: rgba(246, 199, 84, 0.1);

}

### **2. Button Components**

#### **Primary Button**

const PrimaryButton = ({ children, onClick, disabled, loading, size = 'medium' }) => {

const sizeClasses = {

small: 'btn-small',

medium: 'btn-medium',

large: 'btn-large'

};

return (

<button

className={`btn btn-primary ${sizeClasses[size]} ${loading ? 'btn-loading' : ''}`}

onClick={onClick}

disabled={disabled || loading}

>

{loading ? <Spinner size="small" color="white" /> : children}

</button>

);

};

.btn {

font-family: 'Geist', sans-serif;

font-weight: 500;

border: none;

border-radius: 8px;

cursor: pointer;

transition: all 0.2s ease;

display: inline-flex;

align-items: center;

justify-content: center;

gap: 8px;

outline: none;

}

.btn-primary {

background: var(--iris);

color: var(--white);

}

.btn-primary:hover:not(:disabled) {

background: #5A3FE5;

transform: translateY(-1px);

}

.btn-primary:active:not(:disabled) {

transform: translateY(0);

}

.btn-primary:disabled {

opacity: 0.5;

cursor: not-allowed;

}

/\* Size variants \*/

.btn-small {

padding: 8px 16px;

font-size: 14px;

min-height: 32px;

}

.btn-medium {

padding: 12px 24px;

font-size: 16px;

min-height: 44px;

}

.btn-large {

padding: 16px 32px;

font-size: 18px;

min-height: 56px;

}

#### **Secondary Button**

.btn-secondary {

background: transparent;

color: var(--iris);

border: 1.5px solid var(--iris);

}

.btn-secondary:hover:not(:disabled) {

background: var(--iris-light);

}

#### **Danger Button**

.btn-danger {

background: var(--rose);

color: var(--white);

}

.btn-danger:hover:not(:disabled) {

background: #E54766;

}

### **3. Card Component**

const Card = ({ title, subtitle, children, actions, status }) => {

return (

<div className={`card ${status ? `card-${status}` : ''}`}>

{(title || subtitle) && (

<div className="card-header">

{title && <h3 className="card-title">{title}</h3>}

{subtitle && <p className="card-subtitle">{subtitle}</p>}

</div>

)}

<div className="card-content">

{children}

</div>

{actions && (

<div className="card-actions">

{actions}

</div>

)}

</div>

);

};

.card {

background: var(--white);

border-radius: 12px;

box-shadow: 0 1px 3px rgba(7, 9, 33, 0.06),

0 4px 6px rgba(7, 9, 33, 0.04);

transition: box-shadow 0.2s ease;

overflow: hidden;

}

.card:hover {

box-shadow: 0 4px 6px rgba(7, 9, 33, 0.08),

0 10px 15px rgba(7, 9, 33, 0.06);

}

.card-header {

padding: 24px 24px 16px;

border-bottom: 1px solid var(--ink-10);

}

.card-title {

font-family: 'Geist', sans-serif;

font-size: 20px;

font-weight: 500;

color: var(--ink);

margin: 0;

}

.card-subtitle {

font-family: 'Geist', sans-serif;

font-size: 14px;

color: var(--ink-60);

margin: 4px 0 0;

}

.card-content {

padding: 24px;

}

.card-actions {

padding: 16px 24px;

background: var(--fog);

border-top: 1px solid var(--ink-10);

display: flex;

justify-content: flex-end;

gap: 12px;

}

/\* Status variants \*/

.card-active {

border-left: 4px solid var(--iris);

}

.card-warning {

border-left: 4px solid var(--sun);

}

.card-danger {

border-left: 4px solid var(--rose);

}

### **4. Form Components**

#### **Input Field**

const InputField = ({ label, value, onChange, error, helper, type = 'text', required }) => {

return (

<div className="form-group">

{label && (

<label className="form-label">

{label} {required && <span className="required">\*</span>}

</label>

)}

<input

type={type}

className={`form-input ${error ? 'form-input-error' : ''}`}

value={value}

onChange={onChange}

/>

{helper && !error && <span className="form-helper">{helper}</span>}

{error && <span className="form-error">{error}</span>}

</div>

);

};

.form-group {

margin-bottom: 24px;

}

.form-label {

display: block;

font-family: 'Geist', sans-serif;

font-size: 14px;

font-weight: 500;

color: var(--ink);

margin-bottom: 8px;

}

.required {

color: var(--rose);

margin-left: 2px;

}

.form-input {

width: 100%;

padding: 12px 16px;

font-family: 'Geist', sans-serif;

font-size: 16px;

color: var(--ink);

background: var(--white);

border: 1.5px solid var(--ink-20);

border-radius: 8px;

transition: border-color 0.2s ease;

outline: none;

}

.form-input:focus {

border-color: var(--iris);

box-shadow: 0 0 0 3px var(--iris-light);

}

.form-input-error {

border-color: var(--rose);

}

.form-input-error:focus {

box-shadow: 0 0 0 3px var(--rose-light);

}

.form-helper {

display: block;

font-family: 'Geist', sans-serif;

font-size: 13px;

color: var(--ink-60);

margin-top: 4px;

}

.form-error {

display: block;

font-family: 'Geist', sans-serif;

font-size: 13px;

color: var(--rose);

margin-top: 4px;

}

#### **Select Dropdown**

const Select = ({ label, value, onChange, options, placeholder }) => {

return (

<div className="form-group">

{label && <label className="form-label">{label}</label>}

<div className="select-wrapper">

<select className="form-select" value={value} onChange={onChange}>

{placeholder && <option value="">{placeholder}</option>}

{options.map(opt => (

<option key={opt.value} value={opt.value}>{opt.label}</option>

))}

</select>

<ChevronDownIcon className="select-icon" />

</div>

</div>

);

};

.select-wrapper {

position: relative;

}

.form-select {

width: 100%;

padding: 12px 40px 12px 16px;

font-family: 'Geist', sans-serif;

font-size: 16px;

color: var(--ink);

background: var(--white);

border: 1.5px solid var(--ink-20);

border-radius: 8px;

appearance: none;

cursor: pointer;

transition: border-color 0.2s ease;

outline: none;

}

.form-select:focus {

border-color: var(--iris);

box-shadow: 0 0 0 3px var(--iris-light);

}

.select-icon {

position: absolute;

right: 16px;

top: 50%;

transform: translateY(-50%);

width: 20px;

height: 20px;

color: var(--ink-60);

pointer-events: none;

}

### **5. Navigation Components**

#### **Sidebar Navigation**

const SidebarNav = ({ items, activeItem, onItemClick }) => {

return (

<nav className="sidebar-nav">

{items.map(item => (

<button

key={item.id}

className={`nav-item ${activeItem === item.id ? 'nav-item-active' : ''}`}

onClick={() => onItemClick(item.id)}

>

<span className="nav-icon">{item.icon}</span>

<span className="nav-label">{item.label}</span>

</button>

))}

</nav>

);

};

.sidebar-nav {

width: 280px;

background: var(--white);

border-right: 1px solid var(--ink-10);

padding: 24px 16px;

height: 100vh;

overflow-y: auto;

}

.nav-item {

width: 100%;

display: flex;

align-items: center;

gap: 12px;

padding: 12px 16px;

font-family: 'Geist', sans-serif;

font-size: 15px;

font-weight: 400;

color: var(--ink-80);

background: transparent;

border: none;

border-radius: 8px;

cursor: pointer;

transition: all 0.2s ease;

text-align: left;

margin-bottom: 4px;

}

.nav-item:hover {

background: var(--fog);

}

.nav-item-active {

background: var(--iris-light);

color: var(--iris);

font-weight: 500;

}

.nav-icon {

width: 20px;

height: 20px;

display: flex;

align-items: center;

justify-content: center;

}

.nav-label {

flex: 1;

}

### **6. Table Component**

const DataTable = ({ columns, data, onRowClick }) => {

return (

<div className="table-container">

<table className="data-table">

<thead>

<tr>

{columns.map(col => (

<th key={col.key} className="table-header">

{col.label}

</th>

))}

</tr>

</thead>

<tbody>

{data.map((row, index) => (

<tr

key={row.id}

className="table-row"

onClick={() => onRowClick && onRowClick(row)}

>

{columns.map(col => (

<td key={col.key} className="table-cell">

{row[col.key]}

</td>

))}

</tr>

))}

</tbody>

</table>

</div>

);

};

.table-container {

width: 100%;

overflow-x: auto;

background: var(--white);

border-radius: 12px;

box-shadow: 0 1px 3px rgba(7, 9, 33, 0.06);

}

.data-table {

width: 100%;

border-collapse: collapse;

}

.table-header {

font-family: 'Geist', sans-serif;

font-size: 13px;

font-weight: 600;

text-transform: uppercase;

letter-spacing: 0.05em;

color: var(--ink-60);

text-align: left;

padding: 16px;

background: var(--fog);

border-bottom: 1px solid var(--ink-10);

}

.table-row {

transition: background 0.2s ease;

cursor: pointer;

}

.table-row:hover {

background: var(--fog);

}

.table-row:not(:last-child) {

border-bottom: 1px solid var(--ink-10);

}

.table-cell {

font-family: 'Geist', sans-serif;

font-size: 15px;

color: var(--ink);

padding: 16px;

}

### **7. Progress Components**

#### **Progress Bar**

const ProgressBar = ({ value, max = 100, label, color = 'iris' }) => {

const percentage = (value / max) \* 100;

return (

<div className="progress-container">

{label && <span className="progress-label">{label}</span>}

<div className="progress-bar">

<div

className={`progress-fill progress-${color}`}

style={{ width: `${percentage}%` }}

/>

</div>

<span className="progress-value">{value}/{max}</span>

</div>

);

};

.progress-container {

width: 100%;

}

.progress-label {

display: block;

font-family: 'Geist', sans-serif;

font-size: 14px;

font-weight: 500;

color: var(--ink);

margin-bottom: 8px;

}

.progress-bar {

height: 8px;

background: var(--ink-10);

border-radius: 4px;

overflow: hidden;

}

.progress-fill {

height: 100%;

border-radius: 4px;

transition: width 0.3s ease;

}

.progress-iris {

background: var(--iris);

}

.progress-rose {

background: var(--rose);

}

.progress-sun {

background: var(--sun);

}

.progress-value {

display: inline-block;

font-family: 'Geist Mono', monospace;

font-size: 12px;

color: var(--ink-60);

margin-top: 4px;

}

### **8. Modal Component**

const Modal = ({ isOpen, onClose, title, children, actions }) => {

if (!isOpen) return null;

return (

<div className="modal-overlay" onClick={onClose}>

<div className="modal-content" onClick={e => e.stopPropagation()}>

<div className="modal-header">

<h3 className="modal-title">{title}</h3>

<button className="modal-close" onClick={onClose}>

<CloseIcon />

</button>

</div>

<div className="modal-body">

{children}

</div>

{actions && (

<div className="modal-actions">

{actions}

</div>

)}

</div>

</div>

);

};

.modal-overlay {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(7, 9, 33, 0.4);

display: flex;

align-items: center;

justify-content: center;

z-index: 1000;

animation: fadeIn 0.2s ease;

}

.modal-content {

background: var(--white);

border-radius: 16px;

width: 90%;

max-width: 560px;

max-height: 90vh;

overflow: hidden;

display: flex;

flex-direction: column;

animation: slideUp 0.2s ease;

}

.modal-header {

padding: 24px;

border-bottom: 1px solid var(--ink-10);

display: flex;

align-items: center;

justify-content: space-between;

}

.modal-title {

font-family: 'Geist', sans-serif;

font-size: 20px;

font-weight: 500;

color: var(--ink);

margin: 0;

}

.modal-close {

width: 32px;

height: 32px;

display: flex;

align-items: center;

justify-content: center;

background: transparent;

border: none;

border-radius: 8px;

cursor: pointer;

transition: background 0.2s ease;

}

.modal-close:hover {

background: var(--fog);

}

.modal-body {

padding: 24px;

overflow-y: auto;

flex: 1;

}

.modal-actions {

padding: 24px;

border-top: 1px solid var(--ink-10);

display: flex;

justify-content: flex-end;

gap: 12px;

}

@keyframes fadeIn {

from { opacity: 0; }

to { opacity: 1; }

}

@keyframes slideUp {

from {

opacity: 0;

transform: translateY(20px);

}

to {

opacity: 1;

transform: translateY(0);

}

}

### **9. Badge Component**

const Badge = ({ children, variant = 'default', size = 'medium' }) => {

return (

<span className={`badge badge-${variant} badge-${size}`}>

{children}

</span>

);

};

.badge {

display: inline-flex;

align-items: center;

font-family: 'Geist', sans-serif;

font-weight: 500;

border-radius: 6px;

white-space: nowrap;

}

/\* Size variants \*/

.badge-small {

padding: 2px 8px;

font-size: 11px;

}

.badge-medium {

padding: 4px 10px;

font-size: 12px;

}

.badge-large {

padding: 6px 12px;

font-size: 14px;

}

/\* Color variants \*/

.badge-default {

background: var(--ink-10);

color: var(--ink);

}

.badge-iris {

background: var(--iris-light);

color: var(--iris);

}

.badge-rose {

background: var(--rose-light);

color: var(--rose);

}

.badge-sun {

background: var(--sun-light);

color: #C89D2C; /\* Darker sun for contrast \*/

}

### **10. Toast Notification**

const Toast = ({ message, type = 'info', duration = 3000, onClose }) => {

useEffect(() => {

const timer = setTimeout(onClose, duration);

return () => clearTimeout(timer);

}, [duration, onClose]);

return (

<div className={`toast toast-${type}`}>

<span className="toast-message">{message}</span>

<button className="toast-close" onClick={onClose}>

<CloseIcon />

</button>

</div>

);

};

.toast {

display: flex;

align-items: center;

justify-content: space-between;

gap: 12px;

padding: 16px;

min-width: 320px;

background: var(--white);

border-radius: 8px;

box-shadow: 0 4px 12px rgba(7, 9, 33, 0.12);

animation: slideInRight 0.3s ease;

}

.toast-message {

font-family: 'Geist', sans-serif;

font-size: 14px;

color: var(--ink);

flex: 1;

}

.toast-close {

width: 24px;

height: 24px;

display: flex;

align-items: center;

justify-content: center;

background: transparent;

border: none;

border-radius: 4px;

cursor: pointer;

color: var(--ink-60);

}

.toast-close:hover {

background: var(--fog);

}

/\* Type variants \*/

.toast-success {

border-left: 4px solid var(--iris);

}

.toast-error {

border-left: 4px solid var(--rose);

}

.toast-warning {

border-left: 4px solid var(--sun);

}

.toast-info {

border-left: 4px solid var(--ink-20);

}

@keyframes slideInRight {

from {

opacity: 0;

transform: translateX(100%);

}

to {

opacity: 1;

transform: translateX(0);

}

}

## **User Stories by Module**

### **Module 1: RFQ Creation**

#### **Epic: Business User Creates Insurance RFQ**

**As a** business owner  
 **I want to** create a Request for Quote for my insurance needs  
 **So that I** can get competitive quotes from multiple insurers

##### **User Story 1.1: First-Time RFQ Access**

**As a** first-time user  
 **I want to** understand what insurance product I'm applying for  
 **So that I** can make an informed decision

**Acceptance Criteria:**

* User sees a clean introduction screen with product overview
* No insurance jargon is used in the description
* Key benefits are presented in 3-4 bullet points
* "Begin" button is prominently displayed with Iris color
* Loading time is under 2 seconds

##### **User Story 1.2: Multi-Step Form Completion**

**As a** business user  
 **I want to** fill out the RFQ form with guidance  
 **So that I** can provide accurate information

**Acceptance Criteria:**

* Form is divided into logical steps with progress indicator
* Each field has contextual help in the right panel
* Guidance updates dynamically based on focused field
* Form auto-saves every 30 seconds
* User can navigate between steps without losing data
* Required fields are clearly marked with red asterisk
* Validation messages appear inline without page refresh

##### **User Story 1.3: Dynamic Field Enhancement**

**As a** system administrator  
 **I want to** automatically enhance form fields using AI  
 **So that** users get intelligent guidance

**Acceptance Criteria:**

* CSV templates are processed through Claude Sonnet API
* Field guidance is generated within 2 seconds
* Guidance text is jargon-free and contextual
* System caches guidance for 24 hours
* Batch processing handles 10 templates simultaneously

##### **User Story 1.4: RFQ Submission**

**As a** business user  
 **I want to** review and submit my completed RFQ  
 **So that I** can proceed to distribution

**Acceptance Criteria:**

* Review screen shows all entered information
* User can edit any section before submission
* PDF preview is generated in real-time
* Submission confirmation shows with success animation
* Unique RFQ ID is generated and displayed
* User is automatically redirected to distribution screen

### **Module 2: Bid Distribution**

#### **Epic: Distribute RFQ to Insurance Market**

##### **User Story 2.1: Add Direct Contacts**

**As a** business user  
 **I want to** add my existing broker/insurer contacts  
 **So that** they receive my RFQ

**Acceptance Criteria:**

* Email validation happens in real-time
* Multiple emails can be added via comma separation
* Bulk import accepts CSV files
* Added contacts appear as chips with remove option
* Duplicate emails are automatically detected
* Counter shows total contacts added

##### **User Story 2.2: Select from Sanctuari Network**

**As a** business user  
 **I want to** choose from verified insurers and brokers  
 **So that I** expand my quote options

**Acceptance Criteria:**

* Partner cards show logo, name, rating, and specialization
* Search filters by name, category, and rating
* Multi-select with visual feedback (Iris checkbox)
* Selected count updates in real-time
* Profile preview available on hover/click
* Eligibility is clearly indicated

##### **User Story 2.3: Generate Unique Links**

**As a** system  
 **I want to** create unique bidder links  
 **So that** each bidder has secure access

**Acceptance Criteria:**

* Unique URL generated for each selected bidder
* Links include RFQ view and download capabilities
* Links have configurable expiration dates
* System tracks link access and activity
* Email notifications sent from bids@sanctuari.io

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

#### **Epic: Configure Bid Parameters**

##### **User Story 3.1: Set Submission Deadlines**

**As a** business user  
 **I want to** set quote submission deadlines  
 **So that** bidders know the timeline

**Acceptance Criteria:**

* Calendar widget for date selection
* Time zone handling is automatic
* Deadline prominently displayed to bidders
* Links auto-deactivate after deadline
* Warning shown if deadline is less than 48 hours
* Can extend deadline with notification to bidders

##### **User Story 3.2: Configure Email Workflows**

**As a** business user  
 **I want to** set up automated reminders  
 **So that** bidders stay engaged

**Acceptance Criteria:**

* Pre-built email templates available
* Rich text editor for customization
* Preview shows actual email appearance
* CC/BCC fields configurable
* Trigger conditions clearly defined
* Test email functionality available
* Workflow status visible in dashboard

##### **User Story 3.3: Define Bid Requirements**

**As a** business user  
 **I want to** set specific requirements for quotes  
 **So that** bidders understand my needs

**Acceptance Criteria:**

* Requirements text editor with formatting
* Requirements visible on bidder portal
* Can mark requirements as mandatory
* Update notifications sent to all bidders
* Version history maintained

### **Module 4: Bid Submission Portal (Bidder Interface)**

#### **Epic: Insurance Partners Submit Quotes**

##### **User Story 4.1: View RFQ Details**

**As an** insurer/broker  
 **I want to** view the complete RFQ  
 **So that I** can prepare an accurate quote

**Acceptance Criteria:**

* Clean, readable layout of requirements
* PDF download option available
* Print-friendly version accessible
* Requirements clearly categorized
* Deadline prominently displayed
* Contact information visible

##### **User Story 4.2: Submit Quote with Preview**

**As a** broker/insurer  
 **I want to** upload and verify my quote  
 **So that** information is accurate

**Acceptance Criteria:**

* Drag-drop area accepts PDF/CSV files
* Brokers must select insurer before upload
* Multiple quotes can be submitted
* Llama Parse extracts key information
* Preview shows extracted data
* Inline editing available for corrections
* Confirmation required before final submission
* Upload progress bar visible

##### **User Story 4.3: Upload Policy Wordings**

**As an** insurer  
 **I want to** attach policy documents  
 **So that** clients understand coverage details

**Acceptance Criteria:**

* Separate upload section for policy documents
* File size limit clearly stated (e.g., 10MB)
* Multiple document upload supported
* AI analysis triggered automatically
* Processing status visible
* Documents linked to specific quotes

##### **User Story 4.4: Ask Clarifications**

**As a** bidder  
 **I want to** ask questions about requirements  
 **So that I** can provide accurate quotes

**Acceptance Criteria:**

* Question box available on portal
* Questions visible to business user immediately
* Response notifications sent via email
* Thread view for conversation history
* Attachments supported in questions
* Response time tracking visible

### **Module 5: Bid Centre**

#### **Epic: Manage and Analyze Submitted Quotes**

##### **User Story 5.1: Dashboard Overview**

**As a** business user  
 **I want to** see all my RFQs at a glance  
 **So that I** can manage multiple bids efficiently

**Acceptance Criteria:**

* Active bids shown as cards with status
* Color coding by urgency (Sun for urgent)
* Quick stats show response rate
* Search and filter functionality
* Sort by date, status, responses
* One-click access to individual RFQ

##### **User Story 5.2: Quote Comparison**

**As a** business user  
 **I want to** compare quotes side-by-side  
 **So that I** can make informed decisions

**Acceptance Criteria:**

* Tabular view with sticky headers
* Key metrics highlighted
* Differences marked in Sun yellow
* Best value indicated with Iris badge
* Sort by any column
* Hide/show columns as needed
* Export to PDF/Excel
* AI insights panel shows recommendations

##### **User Story 5.3: AI-Powered Analysis**

**As a** business user  
 **I want to** receive intelligent insights  
 **So that I** understand nuances beyond price

**Acceptance Criteria:**

* Multiple AI agents analyze in parallel
* Coverage gaps identified and highlighted
* Terms & conditions summarized
* Risk flags shown in Rose color
* Recommendations ranked by relevance
* Analysis completes within 30 seconds
* Plain language explanations
* Confidence scores displayed

##### **User Story 5.4: Communication Management**

**As a** business user  
 **I want to** communicate with all bidders  
 **So that I** can clarify and update requirements

**Acceptance Criteria:**

* Broadcast message to all bidders
* Individual bidder messaging
* Message templates available
* Read receipts visible
* Attachment support
* Email notifications sent
* Communication history maintained
* Response time metrics shown

##### **User Story 5.5: Automation Monitoring**

**As a** business user  
 **I want to** track automated workflows  
 **So that I** know what's been sent

**Acceptance Criteria:**

* Automation log shows all triggers
* Email delivery status visible
* Failed automations highlighted in Rose
* Manual retry option available
* Edit automation rules mid-process
* Pause/resume functionality
* Performance metrics displayed

### **Module 6: Profile Management**

#### **Epic: Enhance Broker/Insurer Profiles**

##### **User Story 6.1: Automated Profile Enhancement**

**As a** system  
 **I want to** enrich partner profiles with AI  
 **So that** users can make informed selections

**Acceptance Criteria:**

* Profiles processed through Claude Sonnet
* Batch processing of 20 profiles simultaneously
* Key specializations extracted
* Trust indicators generated
* Jargon-free descriptions created
* Searchable tags added
* Cache refreshed every 24 hours
* Processing completes under 5 seconds per profile

##### **User Story 6.2: Profile Display**

**As a** business user  
 **I want to** view detailed partner profiles  
 **So that I** can select the right partners

**Acceptance Criteria:**

* Clean card layout with company info
* Specializations clearly listed
* Ratings and reviews visible
* Past performance metrics shown
* Contact information available
* Response time statistics displayed
* Verified badge for trusted partners

### **Module 7: Payment & Billing**

#### **Epic: Handle Subscription and Payments**

##### **User Story 7.1: Free First Bid**

**As a** new user  
 **I want to** create my first RFQ free  
 **So that I** can try the platform

**Acceptance Criteria:**

* No payment required for first RFQ
* Full feature access on free bid
* Clear messaging about free tier
* Upgrade prompt after completion
* Benefits of paid tier explained

##### **User Story 7.2: Subscription Payment**

**As a** returning user  
 **I want to** pay for additional RFQs  
 **So that I** can continue using the platform

**Acceptance Criteria:**

* Razorpay integration seamless
* Price (₹1,599) clearly displayed
* Multiple payment methods supported
* Invoice generated automatically
* Payment confirmation immediate
* Subscription status visible in profile
* Refund policy clearly stated

### **Module 8: Admin Panel**

#### **Epic: Platform Administration**

##### **User Story 8.1: User Management**

**As an** admin  
 **I want to** manage platform users  
 **So that I** can ensure quality

**Acceptance Criteria:**

* User list with search and filters
* User activity metrics visible
* Ability to suspend/activate accounts
* View user's RFQ history
* Send direct messages to users
* Export user data to CSV

##### **User Story 8.2: Partner Management**

**As an** admin  
 **I want to** manage insurance partners  
 **So that** quality is maintained

**Acceptance Criteria:**

* Add/remove partners from network
* Verify partner credentials
* Update partner profiles
* Monitor partner performance
* Handle partner complaints
* Set partner visibility rules

##### **User Story 8.3: System Monitoring**

**As an** admin  
 **I want to** monitor system health  
 **So that** issues are caught early

**Acceptance Criteria:**

* Real-time system metrics dashboard
* API performance monitoring
* Error logs with filtering
* AI processing queue visible
* Email delivery status
* Database performance metrics
* Alert system for critical issues

## **Mobile-Specific User Stories**

### **Mobile Optimization**

##### **User Story M.1: Mobile RFQ Creation**

**As a** mobile user  
 **I want to** create RFQs on my phone  
 **So that I** can work on-the-go

**Acceptance Criteria:**

* Forms adapt to mobile viewport
* Touch-friendly inputs (min 44px targets)
* Step navigation via swipe gestures
* Keyboard doesn't obscure inputs
* Auto-zoom disabled on focus
* Progress saved across sessions

##### **User Story M.2: Mobile Quote Review**

**As a** mobile user  
 **I want to** review quotes on mobile  
 **So that I** can make decisions anywhere

**Acceptance Criteria:**

* Comparison table scrollable horizontally
* Key metrics in card view option
* Pinch-to-zoom on documents
* Bottom sheet for actions
* Swipe between quotes
* Share functionality native-enabled

## **Performance User Stories**

##### **User Story P.1: Fast Load Times**

**As a** user  
 **I want** pages to load quickly  
 **So that** I can work efficiently

**Acceptance Criteria:**

* First Contentful Paint < 1.5s
* Time to Interactive < 3s
* Lighthouse score > 90
* Image lazy loading implemented
* Code splitting in place
* CDN for static assets

##### **User Story P.2: Real-time Updates**

**As a** user  
 **I want** to see updates instantly  
 **So that** I have current information

**Acceptance Criteria:**

* Supabase subscriptions active
* Updates appear within 100ms
* No page refresh required
* Optimistic UI updates
* Conflict resolution handled
* Offline queue for actions