**Project Design Phase**

**Problem – Solution Fit Template**

|  |  |
| --- | --- |
| **DATE:** | **26-06-2025** |
| **Team ID :** | **LTVIP2025TMID55625** |
| **Project Name :** | **ResolveFlow: Online Complaint Registration and Management System** |

**Instructions:**

* Each section addresses a key problem statement previously identified.
* The "Problem Statement" reiterates the core issue.
* The "Proposed Solution Fit" details how specific components, features, or architectural decisions directly solve that problem.

**1. Problem Statement: Lack of Transparency and Control**

*John lacks real-time, easily accessible information regarding the status and progress of his complaint, leading to uncertainty and a feeling of being uninformed.*

**Proposed Solution Fit:**

* **Frontend (Client-Side) - User Dashboard (tracking):** Provides a centralized interface for John to log in and immediately see the real-time status of his complaints. The UI/UX presentation (using Bootstrap/Material UI) will ensure clarity and ease of access.
* **Backend (Server-Side) - Status Updates:** The backend processes and manages complaint status changes, ensuring the dashboard reflects the most current information.
* **High-Level Data Flow - Real-time Updates:** Socket.io handles bidirectional communication, ensuring that any status changes on the backend are instantly reflected on John's dashboard without manual refresh, providing true "real-time" tracking.
* **External Services/Integrations - Real-time User Notifications:** The backend triggers automated email/SMS notifications for critical status changes (e.g., "Assigned," "Resolved"), proactively informing John and reducing his need to constantly check the system.

**2. Problem Statement: Cumbersome and Inefficient Initial Complaint Submission**

*The traditional methods of initiating a complaint are often perceived as complex and time-consuming, creating an initial barrier for users like John.*

**Proposed Solution Fit:**

* **Frontend (Client-Side) - Complaint Submission:** Dedicated, intuitive forms will guide John through the submission process.
  + **UI/UX Presentation (Bootstrap/Material UI):** Will ensure a user-friendly, multi-step interface with clear guidance and tooltips, simplifying data input.
  + **Drag-and-drop file/image upload:** Streamlined process for attaching evidence.
* **Backend (Server-Side) - Complaint Processing:** The backend is designed to efficiently receive and validate submitted complaint data, ensuring a smooth flow from the user's perspective.
* **High-Level Data Flow - User Interaction & Request to Backend:** The system's design emphasizes a direct and clear path for John's interaction with the frontend, leading to efficient API requests to the backend for submission.

**3. Problem Statement: Communication Gaps and Redundancy**

*John experiences inefficient communication channels where he may need to repeat information, or where direct interaction with a responsible party is not immediate or streamlined.*

**Proposed Solution Fit:**

* **Frontend (Client-Side) - In-app Messaging:** A built-in chat interface provides a direct and centralized communication channel between John and his assigned agent (Sarah).
* **Backend (Server-Side) - Real-time Chat Handling:** Socket.io facilitates low-latency, bidirectional real-time communication for the chat feature, ensuring messages are exchanged instantly.
* **Database - Chat Messages Storage:** All chat history is persistently stored in MongoDB, linked to the specific complaint record. This eliminates redundancy as neither John nor the agent needs to repeat past information.
* **APIs:** Standardized API endpoints ensure consistent and secure exchange of chat messages and related data.

**4. Problem Statement: Concerns Regarding Data Security and Privacy**

*John has underlying concerns about the security and privacy of his personal and complaint-related data when submitting sensitive information.*

**Proposed Solution Fit:**

* **Backend (Server-Side) - Security Enforcement:** The backend implements various security measures:
  + **User Authentication:** Robust mechanisms (hashing, JWTs, potential 2FA) secure John's account.
  + **Authorization (RBAC):** Ensures only authorized personnel (agents/admins) can access specific data.
  + **Input Validation:** Prevents malicious data injection.
  + **Confidentiality Measures:** Design incorporates policies and technical controls to protect sensitive complaint details.
* **Key Non-Functional Considerations - Security:** This entire section is dedicated to addressing security at every layer, from data encryption (in transit via HTTPS, at rest in MongoDB) to API security and regular audits. This proactive approach builds trust and directly mitigates John's privacy concerns.
* **Deployment Strategy (Render):** Leveraging Render provides inherent security benefits like automated SSL certificates for encrypted communication (HTTPS) and robust infrastructure management, reducing the burden on the development team for basic infrastructure security.