1. Solution Requirements

Functional Requirements

These capture the core behaviors of the ResolveNow platform.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**User Registration**	Form-based signup; Email verification link; Social login (Gmail/LinkedIn).
FR-2	**Complaint Management**	Submit complaint with description/attachments; View "My Complaints" status.
FR-3	**Communication**	Real-time chat with assigned agents; Email/SMS notifications for status updates.
FR-4	**Administrative Control**	Intelligent complaint routing; Workload monitoring; Agent assignment.

Non-Functional Requirements

These define the system's quality attributes.

NFR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	User-friendly interface designed with Bootstrap and Material UI for seamless navigation.

NFR No.	Non-Functional Requirement	Description
NFR-2	Security	Data encryption (SSL/TLS); User authentication; Secure password hashing; Access controls.
NFR-3	Reliability	Centralized platform ensuring data persistence via MongoDB and consistent uptime.
NFR-4	Performance	Fast retrieval of complaint data and low-latency real-time messaging via Socket.io.
NFR-5	Availability	Accessible via web browsers 24/7 for global complaint submissions.
NFR-6	Scalability	Microservices-ready client-server model capable of handling high volumes of data.

---

2. Technology Stack

Technical Architecture

ResolveNow follows a **Client-Server Architecture**. The Frontend (React.js) communicates with the Backend (Express/Node.js) via RESTful APIs and maintains real-time updates through Socket.io.

Table-1: Components & Technologies

S.No	Component	Description	Technology
1	User Interface	Responsive Web UI for Customers, Admins, and Agents.	React.js, Bootstrap, Material UI
2	Application Logic	Server-side routing, business logic, and API management.	Node.js, Express.js

S.No	Component	Description	Technology
3	**Real-time Engine**	Bi-directional communication for chat and notifications.	Socket.io
4	**Database**	NoSQL storage for flexible complaint schemas and user profiles.	MongoDB, Mongoose (ODM)
5	**External Interface**	Client-side HTTP requests to interact with the backend.	Axios
6	**Infrastructure**	Local development and version control.	Git/GitHub, VS Code

3. Data Flow & User Stories

User Stories

User Type	Epic	User Story	Acceptance Criteria	Priority
Customer	**Registration**	As a user, I want to create an account so I can track my complaints.	Access to personal dashboard after email verification.	High
Customer	**Submission**	As a user, I want to upload images of defects so the agent understands the issue.	Files are successfully attached to the complaint form.	High
Agent	**Resolution**	As an agent, I want to chat with the user to clarify details.	Message is sent and received in real-time.	Medium

User Type	Epic	User Story	Acceptance Criteria	Priority
Admin	**Monitoring**	As an admin, I want to see all active complaints to assign them efficiently.	View a list of all unassigned tickets.	High

Data Flow Summary

1. **User** submits complaint data via **React UI**.
2. **Express Server** validates the request and stores data in **MongoDB**.
3. **Socket.io** triggers a notification to the **Admin/Agent**.
4. **Agent** updates status, which flows back to the **User** via the dashboard.