**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 26-08-2025 |
| Team ID | LTVIP2025TMID60976 |
| Project Name | **Resolve Now: Your Platform for Online Complaints** |
| Maximum Marks | 4 Marks |

**✅ Technical Architecture – Resolve Now (Online Complaint Resolution Platform)**

**🏗️ Overview:**

**Resolve Now is designed using a scalable 3-tier architecture, including:**

* **Presentation Layer (Frontend): Enables users to file, track, and respond to complaints via a modern, accessible web interface.**
* **Application Layer (Backend): Handles all complaint workflows, messaging, status management, and admin moderation using RESTful APIs.**
* **Data Storage Layer: Stores users, complaints, messages, actions, and status logs using a flexible NoSQL database.**

**The system integrates third-party services such as:**

* **OAuth (Google/GitHub) for login**
* **Email/SMS gateways for notifications**
* **Audit logging for legal and compliance needs**

**📦 Table-1: Components & Technologies**

| **S.No** | **Component** | **Description** | **Technology** |
| --- | --- | --- | --- |
| **1.** | **User Interface** | **Web interface for complainants, respondents, and admins** | **HTML, CSS, JavaScript, React.js, Tailwind CSS** |
| **2.** | **Application Logic-1** | **Complaint submission, tracking, real-time chat** | **Node.js, Express.js** |
| **3.** | **Application Logic-2** | **Admin dashboard, complaint resolution workflows, notifications** | **React.js, Node.js** |
| **4.** | **Database** | **Stores users, complaints, messages, logs, reviews** | **MongoDB** |

**⚙️ Table-2: Application Characteristics**

| **S.No** | **Characteristic** | **Description** | **Technology** |
| --- | --- | --- | --- |
| **5.** | **Open-Source Frameworks** | **Modern frontend/backend frameworks for rapid development** | **React.js, Node.js, Tailwind CSS** |
| **6.** | **Scalable Architecture** | **3-tier MVC-based design, scalable via microservices or containers** | **MVC + RESTful API, Docker/Kubernetes (optional)** |
| **7.** | **Authentication & Security** | **Role-based access, JWT auth, OAuth 2.0 for login** | **JWT, Passport.js, OAuth, HTTPS** |
| **8.** | **Real-Time Communication** | **Enables chat between users and admins** | **Socket.io or Web Sockets** |
| **9.** | **Search & Filters** | **Allow filtering by complaint type, status, or date** | **MongoDB queries + React filters** |
| **10.** | **Notifications** | **Email or SMS alerts for updates and escalations** | **Node Mailer, Twilio, Firebase Cloud Messaging (FCM)** |

**🔐 Security & Compliance Notes:**

* **All sensitive data encrypted in transit (TLS) and at rest (MongoDB with encryption at rest)**
* **Role-based access control: User, Respondent, Admin**
* **Logging and audit trails for complaint resolution actions**
* **Optional: GDPR/consumer rights compliance based on region**

**📚 References / Best Practices**

* **React.js Documentation**
* [**Node.js Best Practices**](https://github.com/goldbergyoni/nodebestpractices?utm_source=chatgpt.com)
* [**How to Draw Useful Technical Architecture Diagrams – Medium**](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d?utm_source=chatgpt.com)
* **MongoDB Security Best Practices**
* **JWT Auth Flow in Express.js**