**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 16 July 2025 |
| Team ID | LTVIP2025TMID59305 |
| Project Name | Resolve Now-Your Platform for Online Complaints |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

**🔧 Tech Stack Overview**

| **Layer** | **Technology Used** | **Purpose** |
| --- | --- | --- |
| Frontend | React.js, HTML5, CSS3, Bootstrap/TailwindCSS | Build responsive and interactive user interfaces |
| Backend | Node.js, Express.js | Handle RESTful API requests and core business logic |
| Database | MongoDB, Mongoose ORM | Store user data, complaints, chat messages, and resolution details |
| Authentication | JWT (JSON Web Token), bcrypt | Secure login, role-based access control, and token validation |
| File Uploads | Multer | Upload complaint attachments and resolution documents |
| Real-time Chat | Socket.io (optional) | Enable live communication between users and agents |
| PDF Generation | pdfkit / html-pdf / jspdf | Generate downloadable complaint summaries or resolution reports |
| API Testing | Postman | Test and validate backend API endpoints |
| Deployment | Vercel (frontend), Render/Heroku (backend), MongoDB Atlas | Host and scale the complete application |
| Version Control | Git & GitHub | Code versioning, collaboration, and deployment workflows |

**🧱 MERN Architecture (High-Level)**

**Frontend (React.js)**

**↓**

**API Calls (Axios/Fetch)**

**↓**

**Backend (Node.js + Express.js)**

**↓**

**Database (MongoDB via Mongoose)**

**🧩 Component-wise Breakdown**

**🔹 Frontend (React.js)**

* **Built as a Single Page Application (SPA)**
* **Components for:**
  + **Home, Login/Register**
  + **User Dashboard (User/Agent/Admin)**
  + **Complaint Form, Complaint Status Tracker**
  + **Chat Interface, Feedback Form**
* **Form Validation: Using React Hook Form / Formik**
* **Routing: Managed via React Router DOM**

**🔹 Backend (Node.js + Express.js)**

* **RESTful APIs with modular routes and controllers**
* **Middleware:**
  + **authMiddleware.js – Role-based access control (JWT)**
  + **errorHandler.js – Consistent error responses**
* **Main Endpoints:**
  + **/api/auth – User login, registration**
  + **/api/complaints – CRUD operations for complaints**
  + **/api/chat – Messaging between user and agent**
  + **/api/admin – Admin functions: assignment, user management**
  + **/api/feedback – Capture user feedback after resolution**

**🔹 Database (MongoDB + Mongoose)**

* **Collections:**
  + **users – User info (complainants, agents, admins)**
  + **complaints – Complaint details, category, status**
  + **messages – Chat logs between user and agent**
  + **feedback – User ratings/comments post-resolution**
* **Mongoose Schemas: Define relationships, validations, and references**

**🔹 Authentication & Security**

* **JWT-based login, token stored in localStorage**
* **Passwords hashed with bcrypt**
* **Role-based route protection using custom middleware**
* **CORS enabled for secure frontend-backend communication**

**⚙️ Deployment Strategy**

| **Component** | **Platform** | **Purpose** |
| --- | --- | --- |
| **Frontend** | **Vercel** | **Hosting React SPA** |
| **Backend** | **Render / Heroku** | **Hosting Express.js API server** |
| **Database** | **MongoDB Atlas** | **Cloud-based MongoDB storage** |

**🛡️ Additional Tools (Optional but Recommended)**

| **Tool/Library** | **Purpose** |
| --- | --- |
| **Redux / Context API** | **State management (user, complaints, roles)** |
| **Framer Motion** | **Smooth UI transitions and animations** |
| **Cloudinary / Firebase** | **File/image hosting for attachments** |
| **Swagger** | **API documentation and testing interface** |
| **Jest / Mocha** | **Unit and integration testing for APIs** |