# ***Client Feedback and Issue Tracking Portal Works***

## ****Goal of the System****

To provide companies with a **centralized platform** where clients can **submit issues, bugs, or feedback**, and company support or development teams can **track, discuss, and resolve them efficiently** — improving communication, customer satisfaction, and service quality.

## ****Key Actors (Users)****

1. **Client (Customer / End User)**
   * Submits feedback, issues, or feature requests
   * Tracks the progress and status of their requests
   * Communicates with support staff (via comments or chat)
   * Rates resolution and satisfaction after closure
2. **Support Staff / Developer**
   * Views all issues assigned to them
   * Reviews, replies, and resolves issues
   * Updates ticket status (e.g., Open → In Progress → Resolved)
   * Communicates with the client to clarify problems
3. **Manager / Administrator**
   * Monitors all ongoing issues and their resolution progress
   * Assigns tickets to staff
   * Views performance reports and analytics
   * Manages users, roles, and system configurations

## ****System Workflow****

### ****1. Client Registration and Login****

* Clients create an account or log in using company-provided credentials.
* Once logged in, they can access their dashboard to view existing tickets or submit new ones.

**Outcome:** Client account created, ready to report issues.

### ****2. Submitting a New Feedback or Issue****

* Client clicks “Create New Ticket”.
* Fills out a form with:
  + Title (e.g., “Website payment not working”)
  + Description of the problem
  + Type: Bug / Feature Request / General Feedback
  + Priority: Low, Medium, High
  + Optional: attach a screenshot or file
* The system stores this data in the **Issue Table**, sets the **status = “Open”**, and sends an alert to the support team.

**Outcome:** A new “ticket” is created in the database and visible to staff and managers.

### ****3. Staff Review and Assignment****

* Manager or admin views new issues in the dashboard.
* Each issue can be:
  + Assigned to a support staff/developer
  + Prioritized (e.g., “High Priority”)
  + Commented on or updated
* Assigned staff gets notified by email or dashboard alert.

**Outcome:** Ticket officially assigned for handling.

### ****4. Communication Between Client and Staff****

There are **two ways** this communication can work, depending on system complexity:

#### Option A: Comment Thread

* Inside each issue, there’s a **comment section**.
* Both the client and staff can write messages (like GitHub issue comments):
* Client: My payment failed today.
* Staff: Please confirm if you used Visa or MasterCard.
* Client: Visa.
* Staff: We found the error. It’s fixed now — try again.
* Each comment has author, timestamp, and optional attachment.

**Outcome:** Communication is stored and transparent for both parties.

#### Option B: Real-Time Chat (Advanced)

* Uses **WebSockets** (Django Channels + React) for instant messaging.
* Similar to live chat — updates in real time.

**Outcome:** Fast, real-time communication.

### ****5. Issue Status Update****

* Support staff changes ticket status based on progress:
  + **Open** → **In Progress** → **Resolved** → **Closed**
* Each update triggers notifications to the client.

**Outcome:** Both client and manager always know the current state of the issue.

### ****6. Client Confirmation and Rating****

* After resolution, the client gets notified.
* They can confirm whether the issue is truly fixed.
* Optionally give a rating (e.g., ⭐⭐⭐⭐ for satisfaction).
* If not satisfied, the issue can be reopened.

**Outcome:** System collects service quality data for reports.

### ****7. Reports and Analytics (Manager Dashboard)****

Managers can view:

* Total tickets received (by type, priority, and date)
* Open vs. Resolved issues
* Average resolution time
* Staff performance reports
* Client satisfaction statistics

**Outcome:** Data-driven insight for improving service quality.

**8. Administration and Maintenance**

Admin can:

* Manage user accounts and roles
* Configure categories (Bug, Feature, Complaint, etc.)
* Backup and restore data
* Control email templates and notification settings

**Outcome:** Full system control and easy maintenance.

## ****Core Features by User Role****

|  |  |
| --- | --- |
| **Role** | **Features** |
| **Client** | Register, submit issues, comment, view progress, give feedback |
| **Support Staff** | View assigned tickets, communicate, update status, resolve issues |
| **Manager/Admin** | Assign tasks, monitor analytics, manage users, generate reports |

## ****System Flow Summary****

Client → Submits Issue → Manager Assigns → Staff Works & Communicates → Client Confirms → Manager Reviews & Reports

## ****Dashboard Views****

### Client Dashboard

* “Submit New Issue” button
* “My Tickets” list with statuses (Open / In Progress / Resolved)
* “Feedback” and rating forms
* Notification alerts

### Staff Dashboard

* “Assigned Tickets” list
* “Pending Work” panel
* “Add Comment” and “Change Status” buttons
* “Performance Stats” view

### Manager/Admin Dashboard

* “All Tickets Overview”
* “Assign Staff” module
* “Analytics Charts” (resolved vs. pending)
* “Export Reports”
* “User Management” panel

## ****Database Entities****

|  |  |
| --- | --- |
| **Table** | **Description** |
| **User** | Stores all registered clients, staff, and admins |
| **Issue** | Stores issue details (title, description, priority, status, etc.) |
| **Comment** | Holds communication thread for each issue |
| **Attachment** | Optional screenshots or files linked to issue |
| **Feedback** | Stores client ratings after issue resolution |
| **Notification** | Tracks system alerts and messages |

**Tech stack**

**React + Tailwindcss + Django(Django REST Framework)**