

# Project Architecture & Role Responsibilities Document

## Project Context

This project is a **Frontend-Only Insurance Policy Management System** using:

- Angular 20/21 (Standalone)
- Local Storage (acting as database)
- JSON Server (for development/testing)
- Local folder storage for documents & images

The system supports **three main roles**: Customer, Agent, and Admin. Each role has defined responsibilities, UI access, and workflow ownership.

---

## 1. High-Level Architecture

### 1.1 Layers

#### 1. UI Layer (Angular Frontend)

- Role-based dashboards (Customer / Agent / Admin)
- Forms for requests & approvals
- Tables for policy data
- Local file upload for documents/images

#### 2. Service Layer (Angular Services)

- AuthService – role handling & session storage
- PolicyService – CRUD operations
- RequestService – endorsements, renewals, cancellations
- StorageService – localStorage + file references

#### 3. Storage Layer

- localStorage (main data store)
  - JSON Server (mock backend for testing)
  - Local folder (for document/image uploads)
-

## 2. Role-Based Access & Responsibilities

### 2.1 Customer Role

**Purpose:** End user managing their own insurance policies

**Features:**

- View all active policies
- Request policy renewal
- Request policy cancellation
- Upload supporting documents
- Track request status

**Actions Owned by Customer:**

- Renewal Request
- Cancellation Request

**Data Created by Customer:**

- Renewal requests
  - Cancellation requests
  - Uploaded documents
- 

### 2.2 Agent Role

**Purpose:** Intermediate approver handling customer requests

**Features:**

- View assigned customer policies
- View incoming renewal/cancellation requests
- Approve or reject policy endorsements
- Add remarks/comments

**Actions Owned by Agent:**

- Policy endorsement request approval

**Data Created by Agent:**

- Approval status
  - Agent remarks
-

## 2.3 Admin Role

**Purpose:** System owner and policy manager

**Features:**

- Create new insurance policies
- Update existing policy details
- Delete obsolete policies
- View all system requests
- Manage users & roles

**Actions Owned by Admin:**

- CRUD policies

**Data Created by Admin:**

- Policy master records
  - User-role mappings
- 

## 3. End-to-End Workflow

### 3.1 Policy Renewal Flow

1. Customer submits renewal request
  2. Request stored in localStorage
  3. Agent views pending requests
  4. Agent approves/rejects
  5. Status updated & reflected to Customer
- 

### 3.2 Policy Cancellation Flow

1. Customer submits cancellation request
  2. Request stored in localStorage
  3. Agent reviews cancellation
  4. Agent approves/rejects
  5. Final status shown to Customer
-

### 3.3 Policy Endorsement Flow

1. Customer raises endorsement request
  2. Agent reviews changes
  3. Agent approves/rejects
  4. Admin updates master policy (if required)
- 

## 4. Data Models (Simplified)

### 4.1 Policy

```
{  
  id: number,  
  name: string,  
  type: string,  
  premium: number,  
  status: string,  
  createdBy: string  
}
```

### 4.2 Customer policy application request

```
{  
  id: number,  
  policyId: number,  
  customerId: number,  
  type: 'RENEWAL' | 'CANCELLATION' | 'ENDORSEMENT',  
  status: 'PENDING' | 'APPROVED' | 'REJECTED',  
  remarks: string,  
  createdAt: string  
}
```

---

## 5. Security & Validation

- Role-based route guards
- Input validation on forms
- File size/type checks
- Session stored in localStorage

## 6. Deployment & Submission

For Sir (Evaluation Submission):

- Angular Frontend Code
  - JSON Server mock DB
  - Local upload folder
  - Architecture Document (this file)
  - Demo credentials for all roles
- 

## 7. Responsibility Matrix

Role	Can Create	Can View	Can Approve	Can Delete
Customer	Requests	Policies	✗	✗
Agent	Remarks	Requests	✓	✗
Admin	Policies	All	✓	✓

---

## 8. Claim Module – Overview

The Claim Module allows customers to request financial compensation for insured events such as accidents, medical emergencies, or travel issues. Claims are processed through a structured workflow involving Customer submission, Agent verification, and Admin settlement. The system ensures role-based access, document validation, and transparent status tracking throughout the claim lifecycle.

---

## 9. Claim Workflow (High-Level)

### Customer Side

- Raise a new claim
- Upload supporting documents
- Track claim status

### Agent Side

- Review submitted claims
- Verify policy & documents
- Approve, reject, or hold claims

## Admin Side

- Final settlement approval
  - Confirm payout amount
  - Close and archive claims
- 

## 10. Claim Status Lifecycle

CREATED → PENDING → ON-HOLD → APPROVED → SETTLED

↓  
REJECTED

---

## 11. Claim Module Architecture (Layered)

### 11.1 UI Layer

- Claim Submission Form
- Claim History Page
- Agent Claim Review Panel
- Admin Claim Settlement Panel

### 11.2 Service Layer

- ClaimService – create & update claims
- ValidationService – policy & document checks
- FileUploadService – handle attachments
- NotificationService – status updates

### 11.3 Storage Layer

- localStorage (claims data)
  - JSON Server (testing)
  - Local folder (uploaded documents)
-

## 12. Data Models

### 12.1 Claim

```
{
  id: number,
  policyId: number,
  customerId: number,
  claimType: 'HEALTH' | 'VEHICLE' | 'TRAVEL' | 'LIFE',
  incidentDate: string,
  claimAmount: number,
  reason: string,
  documents: string[],
  status: 'CREATED' | 'PENDING' | 'ON-HOLD' | 'APPROVED' | 'REJECTED' | 'SETTLED',
  createdAt: string
}
```

---

### 12.2 Claim Review (Agent Action)

```
{
  id: number,
  claimId: number,
  agentId: number,
  decision: 'APPROVED' | 'REJECTED' | 'ON-HOLD',
  remarks: string,
  reviewedAt: string
}
```

---

### 12.3 Claim Settlement (Admin Action)

```
{
  id: number,
  claimId: number,
  adminId: number,
  settlementAmount: number,
  paymentReference: string,
  settlementDate: string,
  status: 'SETTLED'
}
```

---

## 12.4 Claim Document

```
{
  id: number,
  claimId: number,
  fileName: string,
  fileType: 'PDF' | 'JPG' | 'PNG',
  filePath: string,
  uploadedAt: string
}
```

---

## 13. Validation Rules (Claims)

Rule Type	Description
Policy Status	Must be ACTIVE
Claim Date	Within policy duration
Claim Amount	≤ policy coverage
File Type	PDF / JPG / PNG only
File Size	Max 5MB
Bank Details	Mandatory for settlement

---

## 14. Summary (For Report)

The Claim Module extends the insurance system by enabling structured claim submission, review, and settlement. It follows a role-based workflow where customers initiate claims, agents verify details, and admins finalize payouts. The architecture uses a layered approach with clear data models, ensuring traceability, validation, and smooth end-to-end claim handling.

## 15. Summary

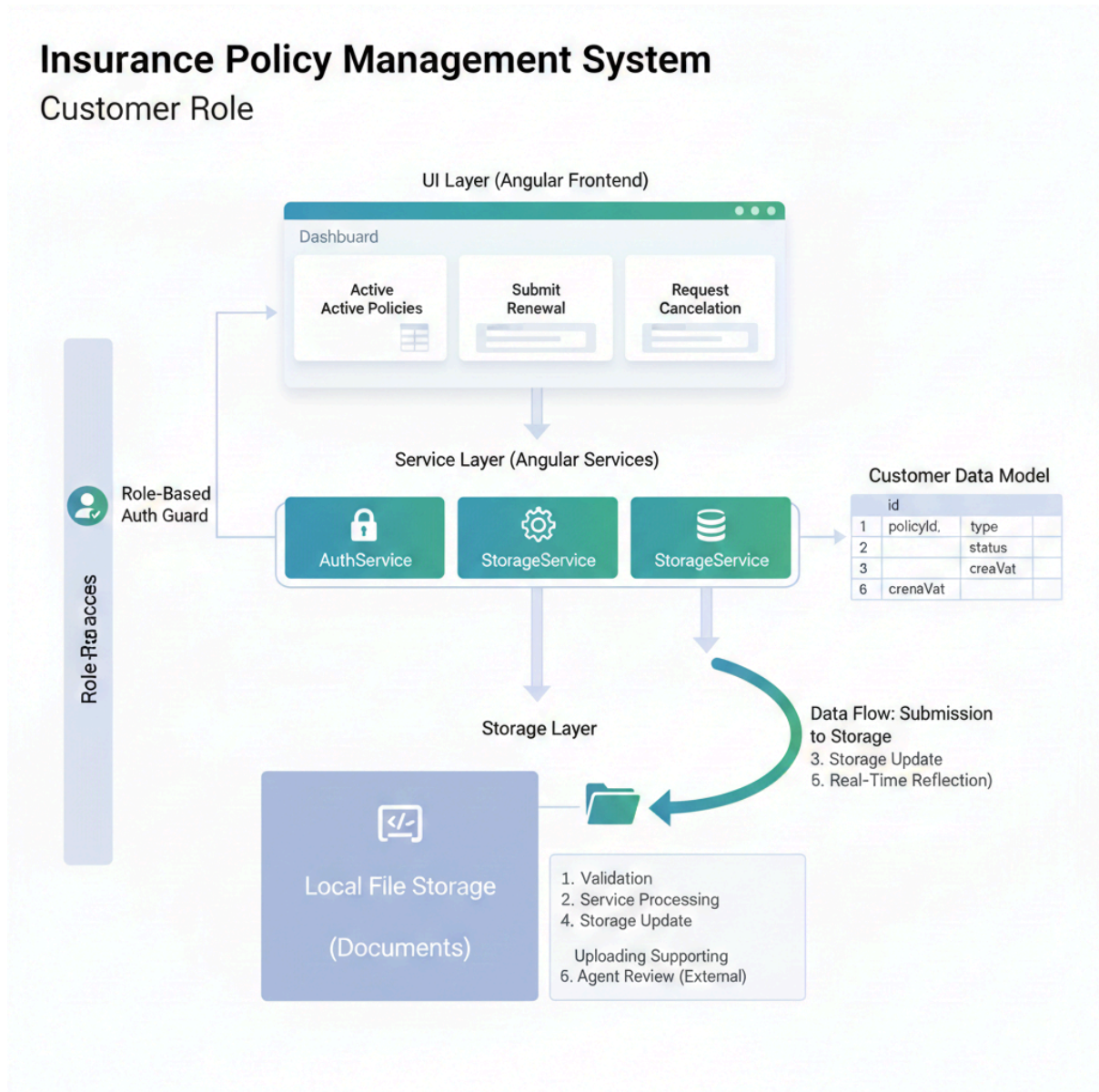
This project demonstrates a **role-based insurance policy management system** built fully on the frontend using Angular and local storage. Each role has clearly separated responsibilities, ensuring clean workflow handling and easy evaluation for academic submission.

## 1. Customer Role Architecture

- **Purpose:** Designed for end-users to manage their personal insurance portfolios.
- **Core Features:** Enables viewing active policies, requesting renewals, and initiating cancellations.
- **Data Interaction:** Customers upload supporting documents and track request statuses in real time.
- **Workflow Ownership:** The customer is the primary owner of Renewal and Cancellation requests.

### Insurance Policy Management System

#### Customer Role



## 2. Agent Role Architecture

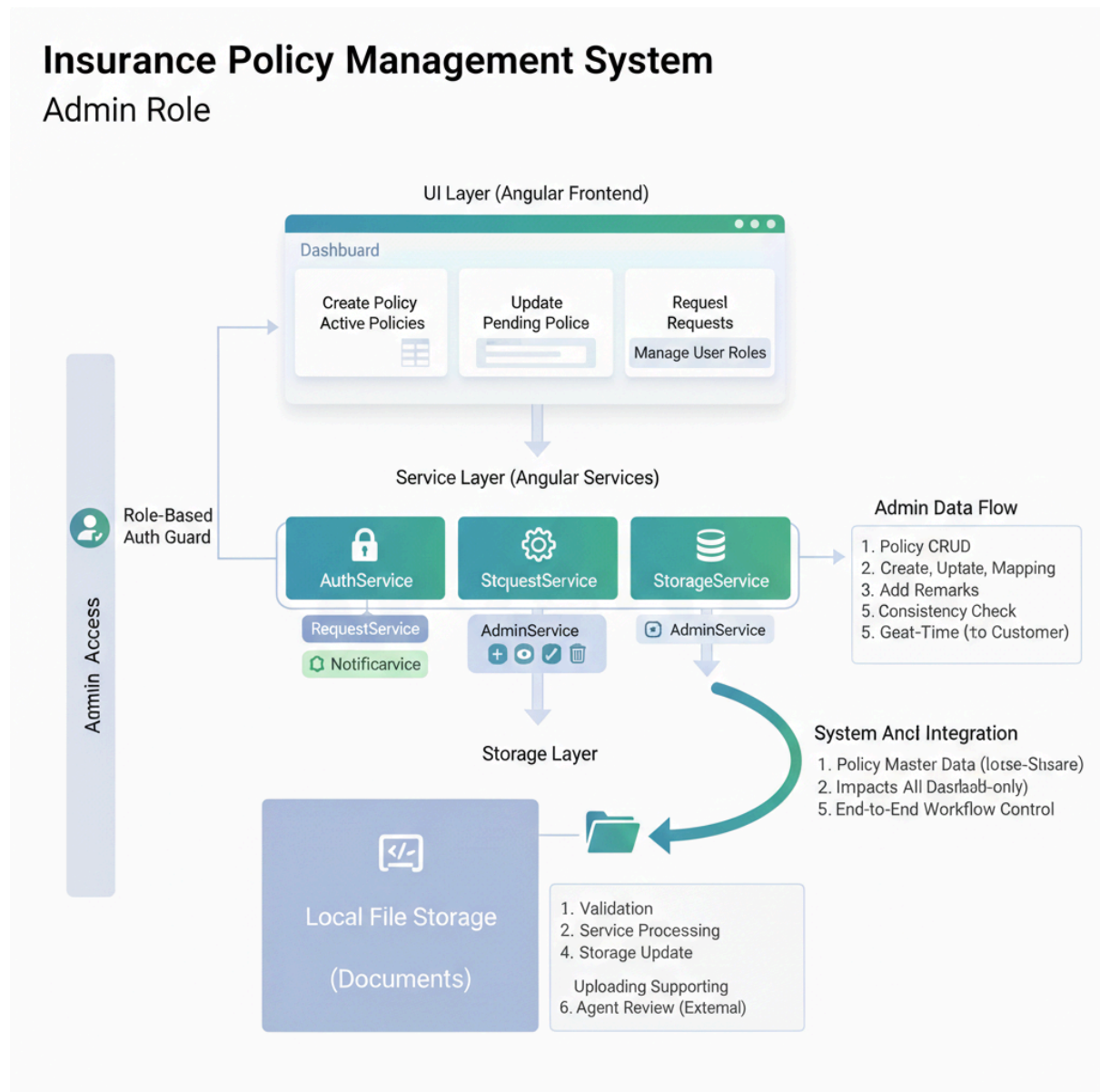
- **Purpose:** Serves as the intermediate approver for customer-initiated workflows.
- **Core Features:** Provides tools to review incoming requests and approve or reject endorsements.
- **Data Interaction:** Agents add remarks or comments to requests, which are persisted in the service layer.
- **Workflow Ownership:** The agent owns the approval status and endorsement review process.

## Admin Role Architecture

- **Purpose:** Acts as the system owner with full control over policy management and user roles.
- **Core Features:** Handles the full CRUD (Create, Read, Update, Delete) lifecycle of insurance policies.
- **Data Interaction:** Manages the master policy records and user-role mappings stored in the storage layer.
- **Workflow Ownership:** Admins update master policies following approved endorsements and oversee all system requests.

# Insurance Policy Management System

## Admin Role



## Team Role Distribution (4 Members)

### Goutham — Authentication & Routing

#### Tasks

- Setup **json-server-auth**
- Create **db.json** users
- Implement **Login & Register UI**
- Handle **JWT storage**
- Create **Auth Guard**
- Create **HTTP Interceptor**
- Role-based redirection (admin / agent / customer)

## Poojitha — Policies & Agent Dashboard

### Tasks

- Create **Policy Catalog UI**
- CRUD operations on policies
- Filter & search policies
- Policy details page
- Admin-only policy management
- JSON Server endpoints for policies
- Agent dashboard

## Srineel — Customer dashboard & Claims

### Tasks

- Customer dashboard

- Claim filing form
- Claim tracking page
- Claim CRUD (create/update status)
- Link customers to policies
- JSON Server claims APIs

## **Sanjay — Admin dashboard & Reports**

### **Tasks**

- Admin dashboard UI
- Show statistics (counts)
- Agent management
- Claims approval page
- Simple reports (tables/charts)
- Dashboard cards (Tailwind)

### **Angular Concepts**

- Data aggregation
- Guards
- Conditional rendering