Software Architecture (SA) for Complaint Management System (CMS)

Overview:

1.1. System Overview

Complaint Management System (CMS) is intended to enable easy accessibility to a shared platform for the users to raise complaints and also effectively monitor and track the lodged complaints thereby prompting necessary action by the management.

1.2. System Context

The system context is defined clearly in the SRS. The user is the source of the complaint. He can make multiple complaints. Resolvers of these complaints - called Admins - will have a list of complaints to look into.

1.3. Stakeholders of CMS

The main stakeholders for the system are the individual users who might use the system to make a complaint, an admin who is supposed to resolve these complaints and the system designer/builder who will build CMS. The main concerns of these stakeholders are:

- For Users: The usability of this platform and reporting issues related to various categories.
- For Admins: The complaint made by the user is to be attended to and will be marked as resolved for further confirmation from the user.
- For designer/builder: The system is easy to modify, particularly to handle future extensions mentioned in the SRS (i.e. the system may be developed to have a 3 tier system of users, admins, and super admins.)

Hence, the key property for which the architecture is to be evaluated is the modifiability or extensibility of the system. Response time performance is another factor for which the system needs to be evaluated.

1.4. Scope of this Document

In this document, we describe two possible architectures for CMS, compare them for various quality attributes, and then choose the most appropriate one, which is our final proposed architecture for CMS. By discussing the two alternatives, we also provide the rationale for selecting the final architecture. For architecture, we consider only the component and connector view.

1.5. Definitions and Acronyms

Definitions :

Complaint: An issue raised by the user pertaining to the departments listed on the platform.

Status: This tells about the response from Admin and whether or not the user approved the response from admin.

Acronyms :

CMS - Complaint Management System

2. Architecture Design

2.1. Architecture 1: The Repository Model

A repository architecture is a system that allows several components to share the same data. Components can be interchanged and are independent of other system components. Each component interfaces the same dataset that is utilized system-wide to perform various tasks.

Below are the components of the architecture.

	Component	**Component Type**	**Description**
1.	Data Repository	Database	This module is the database containing information about the complaints received, user/admin information and the status of a complaint.
2.	Master Controller	Interface Module	This is basically the graphical interface module that interacts with the user on one side and makes appropriate calls to the other modules, discussed below to serve the user requests.
3.	Sign in/Sign out	Processing (Database modification)	This deals with the authentication of individuals trying to access the platform and adds users as and when required.
4.	Lodging Complaints	Processing (Database modification)	This enables the user to make a complaint related to a specified department and notifies the admin. The new complaint is written to the database.

5.	Changing the status of Complaint	Processing (Database modification)	Admin, after responding to a complaint, notifies the complainant regarding the same thus modifying the status quo on the database. The user responds with confirmation re-modifying the status of the same.
6.	Viewing Complaint History	Processing	The list of complaints made/to be attended to will be made available on the user/admin's page respectively.
7.	Account Settings	Processing (Database modification)	If any changes are to be made by the user/admin, they will be able to do so using this module and the changes will be reflected in the database

Below are the connectors of the architecture.

	Connector	**Connector Type**	**Description**
1.	R/W connectors	Database access/ modification	This connector is between modules 3,4,7 and the data repository. These represent access (R) or modification (W) of the data repository.
2.	Read Connectors	Database access	This connector is between the module 6 and the data repository. This connector gets previously lodged complaints from the data repository.
3.	Write Connectors	Database access/ modification	This connector is between module 4 and the data repository. This connector adds complaints to the data repository.
4.	Authentication Connector	HTTP	This connector is between the interface and Google Auth API. This is between the master controller and module 3.
5.	Control Connector	Module invocation or function calls	This connector is between the master controller and the modules 3,4, 5, 6 and 7. This represents the invocation of these modules by the master controller.

Below is the diagram representing the Repository Model.

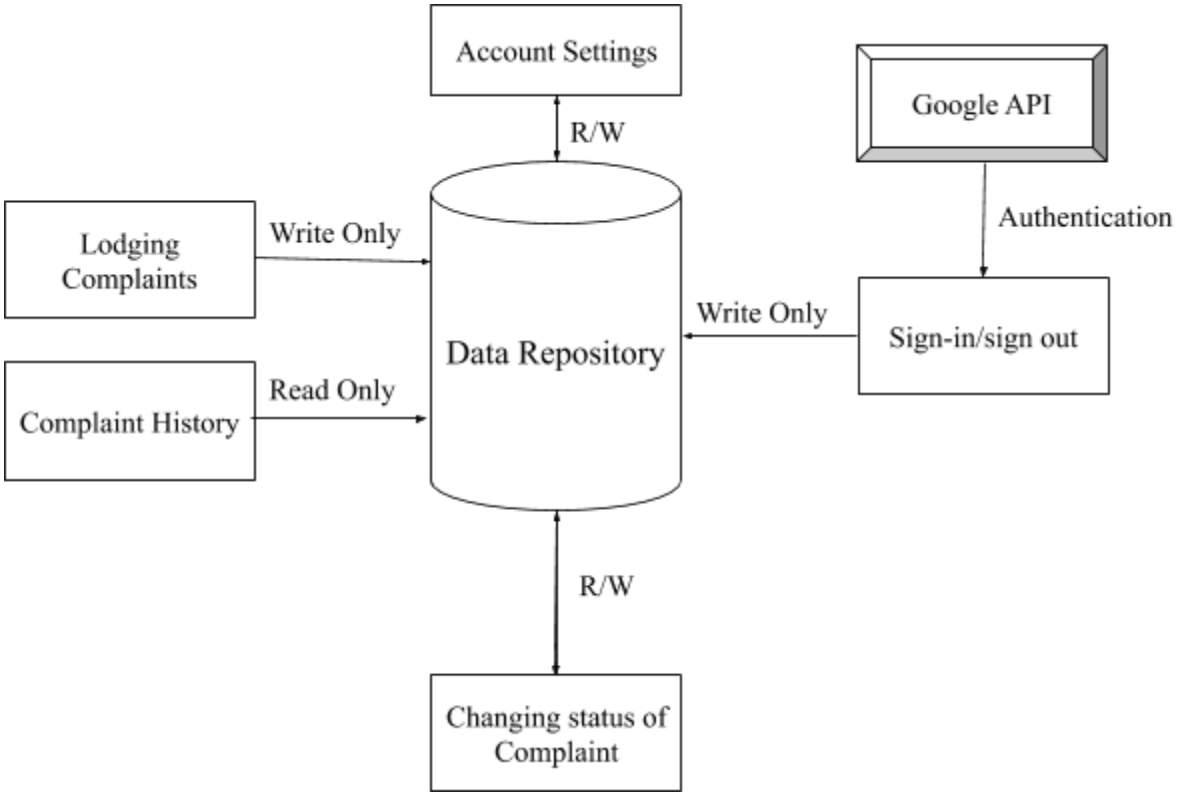

Fig: Repository Model

2.2. Architecture 2: The Access Layer Model (4 layered)

Each layer of the layered architecture pattern has a specific role and responsibility within the application. For example, a presentation layer would be responsible for handling all user interface and browser communication logic, whereas a business layer would be responsible for executing specific business rules associated with the request. Each layer in the architecture forms an abstraction around the work that needs to be done to satisfy a particular business request. For example, the presentation layer doesn't need to know or worry about how to get customer data; it only needs to display that information on a screen in a particular format. Similarly, the business layer doesn't need to be concerned about how to format customer data for display on a screen or even where the customer data is coming from; it only needs to get the data from the persistence layer, perform business logic against the data, and pass that information up to the presentation layer. There is also a component that gets authentication details which takes the user/admin info from the remote database(ie, google authentication server).

Below is the diagram representing the Layer Model.

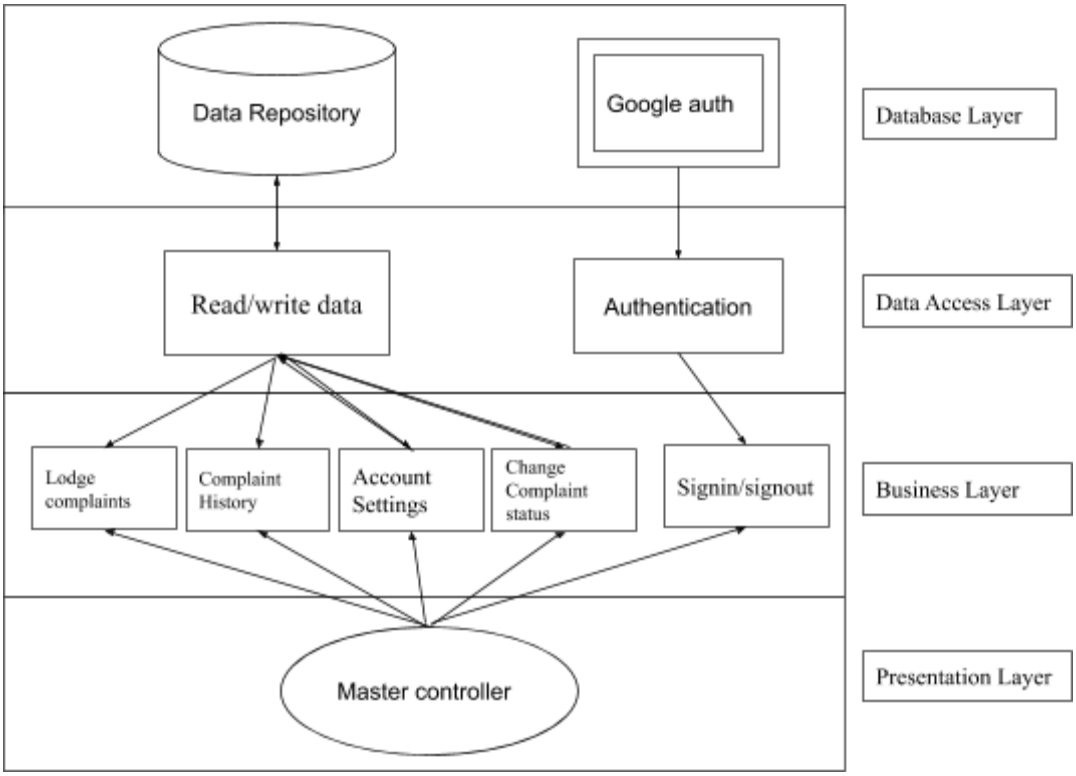

Fig: Layer Model

2.3. Comparing the Architectures

Criteria	Architecture 1	Architecture 2
Change In Data repository	Difficult	Easy
Adapting to changes in authorization	Easy	Easy
Addition in functionalities	Difficult	Easy
Extension to hierarchy in the admin system	Difficult	Easy

3. Final Architecture of CMS

From the above table, we see that architecture 2 is better as far as the change in a data repository is concerned. Moreover, it is also easier to extend it to a multi-level admin system, which involves additional requirements and compartmentalisation of issues. Since in future there may be additional levels of admins added to the system , the functionalities of these admins will be of importance.So , it will be easier to add these when working with layered architecture.Also Architecture 1 better works when components are independent and the inclusion of these co-dependent admin levels to the existing model would be difficult if the said architecture is used. Thus we choose Architecture 2 for the Complaint Management System.

Team Members(Group 07) :

- Gorikapudi Prudhvi - CS16BTECH11016
- Mangu Venkata Sai Subhadra Lakshmi - CS16BTECH11021
- Gonela Deepika - CS16BTECH11015
- Sanivarapu Uma - CS16BTECH11033.