Project Reflection Report
G58: Insurance Hub

1. Our final project aligns with the initial project report we provided. However, there are a few changes. In the initial project report, we mentioned that Insurance Hub would be an insurance aggregator that provides a one-stop solution for users to search, compare, and purchase different insurances. Also, it will have three logins: First, customer login; Second: insurance company login; Third: database administrator login. In stage 5, we demonstrated our project, which provides a digital platform for users to search for insurance policies from major insurance companies. In Insurance Hub, a user (or customer) can search policies, and an insurance company user can add, edit & delete insurance policies. So, we have created two logins, one for the customer and another for the insurance company user. We have not created the database administrator login we proposed in the initial report.

2. In the initial project proposal, we mentioned these points for the usefulness of our project:

A. Our website will filter through insurance policies curated for the customer by taking in various customer inputs.
B. Customers can compare these policies, and we can provide them with additional insights, such as policies with the lowest premium plan.
C. Our website will protect the customer's integrity by encapsulating their data only in the customer's table and not sharing it elsewhere.

We have successfully implemented points A and C. However, we could not implement point B.

3. Our schema is the same as we have mentioned in the initial report. However, we have changed some attributes in our policy table. In the initial report, we mentioned that our policy table would have a premium amount, max discount, and out-of-pocket amount. However, in the ER diagram stage, we have changed these attributes to cover amount, premium per month, and premium per annum. Also, we have added the creation date attribute and removed the features attribute from the policy table.

4. We have not changed anything in our ER diagram or the table implementation. It is the same as we proposed during the initial stage of the project.

5. We have added and removed some functionalities in our project. These are:

Removed functionalities:

- Policy Purchase Option: Initially, we thought of adding a purchase functionality into our application so that a user (or customer) can buy an insurance policy through our application. However, we removed it because we want our application to serve as an aggregator only.

- Database Administrator Login: We have not implemented this functionality because we focused more on customer and insurance company login functionalities.

Added functionalities:

- Pagination: We have added this functionality to our project because, in the development phase of this project, we realized that our database is vast, and it is not good to show 2000-3000 insurance policies on a single page. So, we made some changes in the front-end and back-end of our application to use pagination functionality. Now, a user can search a limited number of policies per page. If they want to check more policies, they can move to the next page.

- Multiple Dashboard Charts: Apart from the filtering policies functionality based on ratings, we have added a world map and a donut chart to our dashboard. In the world map chart, an insurance company user can check the number of policies searched per state by customers. Moreover, in the donut chart, a company user can check the count of different policies. These new features will help the insurance company user to properly analyze the data and gain insights into their published policies.

6. In our application, we have created a dashboard that provides searched policy insights to insurance company users. To implement this dashboard, we have used stored procedures and triggers in the backend of our application which covers our advanced database programs requirement.

7. Technical challenges that we faced are:
- Team member 1 (bachina3): Learning Node.js for writing the server-side code was very new to me. Handling promises and returning responses from SQL to Frontend in Javascript was challenging for me. I learned Express for serving such REST API queries to the client.
- Team member 2 (sharma89): In this project, we have hosted our database on GCP. Before this project, I had not worked on GCP. Learning and using it was new to me. Similarly, working on Sequelize ORM to handle SQL queries through Node.js was a challenge that I learned and overcame.
- Team member 3 (mgoel7): Using JWT tokens to generate one token per user and adding a middleware for all the APIs to make sure that an authenticated user is making the API call was a challenge for me. Handling this server-side logic along with Reactjs development for making Insurance List components was a learning opportunity for me.
- Team member 4 (lpuri2): I integrated session management using cookies and refresh token, which was a new concept for me. Moreover, I also implemented the route guards on the front-end to prevent user access to edit and delete pages for insurance which were only accessible to Company users. Route guards are very useful, and it was a challenging concept for me, but I used Higher Order Components to overcome this challenge.

8. No, whatever changes we have made, we have already explained them in the previous questions.

9. As a future work for this project, we are planning to improve security an

- Security - Right now, the only security our application has is password hashing and the JWT token for authentication. We can improve our application's security by having stronger password policies and parsing the user inputs so that no one can manipulate the data by putting SQL queries in the search bars. We could even implement 2FAs for company and user logins.
- Access Control - Currently, our application is not logically separated between a user and a company. Our application can have two separate services, one serving only User queries and the second serving all the advanced company queries from editing, deleting, and adding more company users for the same company and giving them different levels of access to the Insurance Hub platform.
- Analytics pipeline using NoSQL schema for User Actions - Currently, our User Actions relation does not support multiple types of actions that the user can do. In order to build a complete Analytics pipeline for the Company users to see the activities of normal users, we can integrate a NoSQL schema for user actions with multiple action types, where each action type can have a different value structure.

10. Final division of labor:
    - Project members Lavanya and Meghansh developed the application's front end. They used React for development.
    - Project members Akshit and Rahul developed the application's back end. They used NodeJS for the backend framework, Sequelize as the Node.js ORM and MySQL for the database.
    - The relational schema and database design were developed by all four of us together, where we decided how many entities, and relationships the database will contain.
    - We managed the team work very well. We really liked working with each other. We learned a lot from each other.