Project Reflection Report

Changes in directions of project

No changes were made to the original project proposal.

What the application achieved/failed to achieve regarding its usefulness?

Achievements:

1. Our application GenFlow was made with the goal of providing a generic business workflow management software to cater the needs of small businesses. Generic here means that *with simple configuration, any business owner can configure the application for their business specific use cases, be it a restaurant or a grocery store*. We were able to achieve this by implementing a suitable database schema.
2. Separate user flows for business owners and business process owners, along with different access controls and data abstraction. One example of this is that the business owner could see the aggregate reports, but the process owners couldn't.

Failures:

1. Failure to model non-sequential workflows (not originally in scope): Our application could only handle sequential workflows which was the assumption we made from the inception of the project (keeping in mind the needs of small businesses). However, real life workflows may be more complicated and therefore, we would like to think of this shortcoming as our failure.

Changes to the schema/source of the data

Changes to the schema were made as we moved on the project. These are as follows:

1. Added an attribute called "department" in the Processes Table to track which process is associated with which department.
2. Added CASCADE DELETE functionality so as to remove all related instances across all tables when a workflow instance is deleted.
3. Added auto-increment IDs. Initially we didn't account for how we will be generating the IDs for the various entities present in our tables, and therefore had to make all IDs auto incremental later on.

Changes to the ER Diagram and/or table implementations

No changes were made other than the ones mentioned in the previous question.

What functionalities were added/removed?

The frontend functionality to allow business users to input their business data was not implemented due to the prioritization of tasks and focus on developing a robust

backend/database.
We decided to add reporting graphs for the business owners to understand which departments in their business are lagging behind and need improvement. This was the creative component of our project.

How do advanced DB programs complement the application?

We used a lot of advanced DB functionality in our application including joins, views, trigger, stored procedure, and EXPLAIN ANALYZE. Joins were integral to our database design using which we could find the data related to different workflows and processes for a business. Views were used to simplify the complex queries and make them more readable and easy to debug for the developers. Apart from that, a stored procedure was used for adding a new workflow instance in our database, as there were additional steps to be carried out (like adding the process instance for the first process), and for storing reporting data like the number of pending workflows at each point of time. Trigger was used to keep our database consistent when a user tries to complete the process they have just worked on. This is because several conditional checks and operations have to be performed. Explain Analyze was helpful to identify the indexes required for efficient querying of the data.

Technical challenges faced by each team member

Bharati Kulkarni (bk28) – Ensuring that the statements in the complete procedure are all executed or none executed. It was a very big challenge even while developing as partial execution of the statements would have corrupted the test data. To overcome that, I used transactions. Apart from that, another challenge that we faced was coming up with the database design itself. There were many challenges in the same because of the complexities in the business logic and the constraints for each workflow and process.

Samarth Gupta (sg73) - The data generation across so many tables itself was quite complex, considering that we had to simulate actual workflows and processes in a business. Going forward, the complex queries that we implemented required a lot of debugging because of repeating column names when joining tables, and columns with the same names across different tables. However we were able to sort this out using VIEWS. Lastly, parsing SQL database response which is a relational response into a JSON response suitable for the database was quite challenging.

Ishita Karna(ikarna2) -
Implementing the stored procedure accepting parameters took a lot of effort and time as it was hard to debug and figure out the errors. Syntactically, the procedure would be saved but it failed on some unknown step during its execution. Ultimately, I was able to figure it out by running the stored procedure piece by piece.
Developing the frontend also had challenges of its own. Considering the complexity of the nested data returned by the backend, it was a challenge to efficiently integrate and display this in the frontend. Accurate communication with the database was also necessary to ensure

smooth functioning of the application. Incremental development with focus on details helped me ultimately achieve the desired end result.

Aditya Kumbhar (kumbhar2) -
One of the few challenges that I faced was identifying the right columns/attributes to apply indexes to. The EXPLAIN ANALYZE command was helpful in understanding if an index was speeding up the queries, but some indexes did the opposite of what we expected them to do. So it was tricky to find the right combination of attributes to apply triggers on and required some trial-and-error.
Another challenge I faced was how to get rid of redundant code from the backend that needed to be executed whenever an insert in a specific table occurred. I overcame this challenge by writing a trigger that eliminated the need of programmatic queries to be executed in sequence after an event by moving the redundant DML to the database layer.

Other things that changed

The UI designs planned initially were changed to easy to understand designs in order to balance the workload between frontend , backend and database design.

Future scope

Going forward, we can implement the business owner data input to the application which could consist of all the information about the entire business including the specific workflows and processes in them. We can also add non-sequential workflows in the application which would make the application more versatile.

Final Division of the labor and team management

Bharati Kulkarni (bk28): Responsible for DB designing and REST API development.
Samarth Gupta (sg73): Responsible for DB designing and REST API development.
Ishita Karna (ikarna2): Responsible for DB designing and frontend UI design and development.
Aditya Kumbhar (kumbhar2): Responsible for DB designing, frontend development, and contributed to REST API development.

Team management – Scrum meetings were held every week to understand what each person is doing and discuss any difficulties that the team members are experiencing. Aligning each person’s skills and interests with the work assigned was also ensured. It was difficult to gauge when something would be done. Each person took ownership of the portion allotted to them and thus, finished their chunk in predicted time.