CSCI5408 – FEASIBILITY STUDY REPORT

Group: DP-17

Instructor – Dr. Saurabh Dey

Dhrumil Amish Shah (B00857606)
Jaspreet Kaur Gill (B00879409)
Samiksha Narendra Salgaonkar (B00865423)

Introduction

This project incorporates the design, implementation, testing, and release of a simple relational database management system (RDBMS). This feasibility analysis report describes the data structures used, the flowchart, and the algorithm that will be followed for the implementation. This report also details the project timeline for implementation, along with the initial test cases that have been identified. We have also added the logs and screenshots of the team meetings conducted.

Programming Language

Language: Java

Reason for the choice:

- Java is a versatile language encompassing numerous data structures by exposing the Collection framework APIs that include implementations of hash tables, maps, sets, queues, and lists. These data structures will help us implement the desired Relational Database Management System (RDBMS).
- Java also allows easy implementation of non-linear data structures like Trees.
- Team members have profound knowledge and understanding of underlying concepts of Object-Oriented Programming (OOP), which is advantageous in achieving code efficiency and management.
- Having good hands-on experience in implementing Java solutions, we can use the best practices in coding, testing, and delivering optimal and maintainable solutions.

Data Structures

We plan to implement a Relational Database Management System (RDBMS) using Java Collection APIs that provide the best time complexity with their operations. This would ensure the implementation of efficient in-memory operations for the database. We have concluded on the given collections for the implementation:

Lists: Lists allow the storage of data in a structured format. This interface can be implemented to provide the following advantages:

- Lists provide a time complexity of O(1) in the best-case scenario and O(n) to insert elements in the worst-case scenario.
- It offers a time complexity of O(n) to retrieve elements from the structure except for Array lists that offer O(1).
- The time complexity offered to delete elements from the list is O(n).

Maps: Maps stores data in a key-value format. It can be used by instantiating any sub-classes such as hash map, tree map, linked hash map. It provides the best efficiency for its operations and thus can be used to implement an efficient RDBMS. The performance details are as follows:

- Maps provide a time complexity of O(1) for searching elements which will be advantageous while deleting and updating records in the database.
- It offers a time complexity of O(1) for retrieving elements from the structure.

We plan on implementing a combination of Lists and Maps in our project. For instance, it would be using a list of maps and vice-a-versa.

Flow Chart

We will implement a Relational Database Management System (RDBMS) that will be hosted on a local machine. This database management system is a console-based application. Authenticated users will be able to perform basic DDL, and DML queries to perform operations on the tables. It will also allow users to generate SQL dumps and reverse engineer ERD diagrams from the existing tables in the schema.

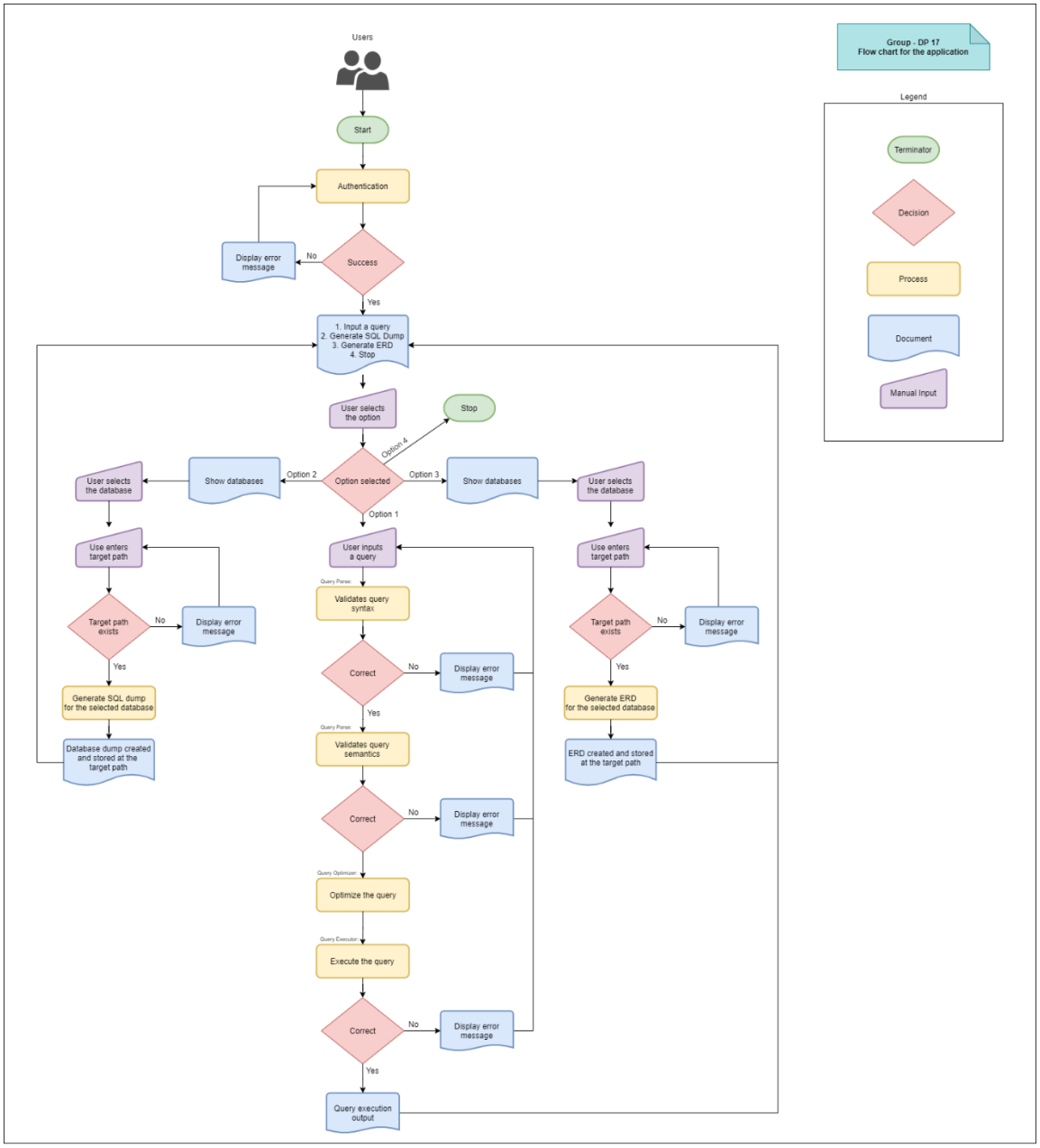

Figure 1: Flowchart for the application

User Authentication
The console will prompt the user to enter a username and password. The user should enter a valid username and password to start using the features of the application. If the user enters an invalid username or password, the interface will show an error message to the user.

Query Syntax Validator
An authenticated user can execute the queries to perform desired operations on the tables in the database. When the user enters the query, the query syntax validator will first validate the syntax's correctness. If the syntax of the query is correct, then the query will be passed for the semantic check; otherwise, a syntax error will be shown to the user via the console.

Query Semantic Validator
The semantic validator will verify the semantics of the query, which means it will validate whether the object names such as database name, table name and column name are correctly mentioned in the query. The semantic validator will verify whether the object names specified in the query exists or not. If semantic validation of the query passes, then the query will be passed to the query optimizer. On the other hand, if the semantic validation of the query fails, then the user will be notified with an error message.

Query Optimizer
The primary function of the query optimizer is to provide an efficient way to execute the query so that the results obtained should be correct and retrieved fast. After optimizing, the query is sent to the query executor for its execution.

Query Executor
During query execution, the query executor will search for the expected details in the data file and then returns the output to the user on the console. For example- The user has executed a query such as SELECT item_name FROM items WHERE item_id=100.
So, the query executor will search the items table in the data file and then search for the record in the table for which item-id is 100 and returns the result to the user. If the user has executed queries to update the table, then permanent changes will be reflected in the table.

SQL Dump Generator
SQL dump generator will allow the user to generate a dump of the schema. The dump contains tables in the database with column names, data types, primary & foreign keys associated with the table columns. The user can select the specific schema for which dump has to be created. The application also allows the user to specify the target path to store the dump. If the specified database and path exist, then the dump will be generated and stored at the target location. However, if the database or path does not exist, the user will be notified via an error message.

ERD Generator
This application also provides a feature to generate an ERD diagram from the existing database. The user must choose the database for which ERD has to be created and the path where the ERD diagram must be stored. ERD will be exported in a file with details like entity names, relationships, primary key, foreign key, and cardinality. ERD will be stored at the mentioned target path if the path and database are correct. However, an error message will be sent to the user if the database does not exist or the provided path is incorrect.

Algorithm

The algorithm for the implementation of the project is described below:

Step 1: Start
Step 2: User performs authentication.
Step 3: If user authentication fails, display an error message, and go to step 2.
Step 4: If user authentication succeeds, go to step 5.
Step 5: Display the below options:
    1. Input a query
    2. Generate SQL Dump
    3. Generate ERD
    4. Stop
Step 6: User selects an option from step 5.
    If option 1 is selected, go to step 7.
    If option 2 is selected, go to step 19.
    If option 3 is selected, go to step 25.
    If option 4 is selected, go to step 31.
Step 7: User inputs a query.
Step 8: Query parser validates the syntax of the query.
Step 9: If query validation fails, display an error message, and go to step 7.
Step 10: If query validation succeeds, go to step 11.
Step 11. Query parser validates the semantics of the query.
Step 12: If query validation fails, display an error message, and go to step 7.
Step 13: If query validation succeeds, go to step 14.
Step 14: Query optimizer optimizes the query.
Step 15: Query executor executes the query.
Step 16: If query execution fails, display an error message, and go to step 7.
Step 17: If query execution succeeds, go to step 18.
Step 18: Display the query execution output and go to step 5.
Step 19: Display all the databases.
Step 20: User selects the database.
Step 21: User enters the target path.
Step 22: If the target path is incorrect, display an error message and go to step 21.
Step 23: If the target path is correct, go to step 24.
Step 24. Generate and store SQL dump at the target path and go to step 5.
Step 25: Display all the databases.
Step 26: User selects the database.
Step 27: User enters the target path.
Step 28: If the target path is incorrect, display an error message and go to step 27.
Step 29: If the target path is correct, go to step 30.
Step 30. Generate and store ERD at the target path and go to step 5.
Step 31: Stop

Test Cases

Test Case 1: Authentication

Test scenario	Login
Test steps	User enters username and password
Expected Results	User logs in successfully with authorization to perform the remaining database operations
Comments	In an event of authentication failure, the system will display an error message and redirect the user for another attempt of authentication

Test Case 2: Query parsing

Test scenario	Verifying syntax of the query
Test steps	User enters a query
Expected Results	It should parse the query successfully without any errors
Comments	In an event of failure, the system will display an error message stating that the query is incorrect and redirect the user for another attempt of entering the query

Test Case 3: Semantic analysis of the query (incorrect database)

Test scenario	Verifying the semantics of the query
Test steps	User enters a valid query (Assumption: Query is parsed successfully)
Expected Results	It should perform the required result based on the query entered such as (create, select, insert, update, drop, delete, truncate)
Comments	In an event of failure, the system will display an error message depending on the error that occurred: If the query contains a database that doesn't exist, then it displays an error stating the same (This is exclusively for all queries except the CREATE query) For CREATE query, if the user enters a database name that already exists, then an error stating the same will be displayed.

Test Case 4: Semantic analysis of the query (incorrect table)

Test scenario	Verifying the semantics of the query
Test steps	User enters a valid query (Assumption: Query is parsed successfully)
Expected Results	It should perform the required result based on the query entered such as (create, select, insert, update, drop, delete, truncate)
Comments	In an event of failure, the system will display an error message depending on the error that occurred: If the query contains a table name that doesn't exist, then it displays an error stating the same (This is exclusively for all queries except the CREATE query) For CREATE query, if the user enters a table name that already exists, then an error stating the same will be displayed.

Test Case 5: CREATE query validation

Test scenario	Verifying the semantics of the CREATE query
Test steps	User enters a valid CREATE query (Assumption: Query is parsed successfully)
Expected Results	It should create a table with the desired attributes and their data types as specified
Comments	In an event of failure, the system will display an error message depending on the error that occurred: If the query contains an unacceptable data type for the attribute to be created within that table, the user will be displayed with an appropriate error message stating the same

Test Case 6: INSERT/UPDATE query validation

Test scenario	Verifying the semantics of the INSERT/UPDATE query
Test steps	User enters a valid INSERT/UPDATE query (Assumption: Query is parsed successfully)
Expected Results	It should insert a row/update a column in the table based on the query executed
Comments	In an event of failure, the system will display an error message stating depending on the error that occurred: If the query contains data in an unacceptable data type for the attribute within that table, the user will be displayed with an appropriate error message stating the same

Test Case 7: SQL Dump target path verification

Test scenario	Verifying the target path specified by the user for SQL Dump creation
Test steps	User enters a valid target path where SQL Dump will be created
Expected Results	It should create a SQL dump at the specified path
Comments	In an event of failure, the system will display an error message stating that the target path doesn't exist

Test Case 8: ERD target path verification

Test scenario	Verifying the target path specified by the user for ERD creation
Test steps	User enters a valid target path where ERD will be created
Expected Results	It should create an ERD of the database at the specified path
Comments	In an event of failure, the system will display an error message stating that the target path doesn't exist

Assumptions

The project will focus on implementing basic SQL queries such as Select, Create, Insert, Update, Drop and will not cover Joins, Nested Queries, Sub Queries, Aggregate functions.

Gantt Chart

Table-1: Task Timeline

Task	Start Date	End Date	Duration
Requirement Gathering Phase	10-06-2021	13-06-2021	3
Knowledge Transfer	13-06-2021	16-06-2021	2
Project Setup	16-06-2021	18-06-2021	2
Design	18-06-2021	21-06-2021	3
Implement Query Parser	21-06-2021	25-06-2021	4
Implement User Authenticator	25-06-2021	30-06-2021	5
Implement Query Executor	30-06-2021	07-07-2021	7
Implement SQL Dump Generator	07-07-2021	14-07-2021	7
Implement ERD Generator	14-07-2021	21-07-2021	7
Miscellaneous Tasks	21-07-2021	25-07-2021	4
Testing	25-07-2021	30-07-2021	5
Release	30-07-2021	01-08-2021	2


Project Timeline
6/10/2021
6/20/2021
6/30/2021
7/10/2021
7/20/2021
7/30/2021
Requirement Gathering Phase
Knowledge Transfer
Project Setup
Design
Implement Query Parser
Implement User Authenticator
Implement Query Executor
Implement SQL Dump Generator
Implement ERD Generator
Miscellaneous Tasks
Testing
Release

Meeting Logs

Meeting 1 – Introduction | Meet & Greet

Figure 2 and 3 displays the meeting logs and Minutes of the meeting (MoM) of our first meeting respectively.

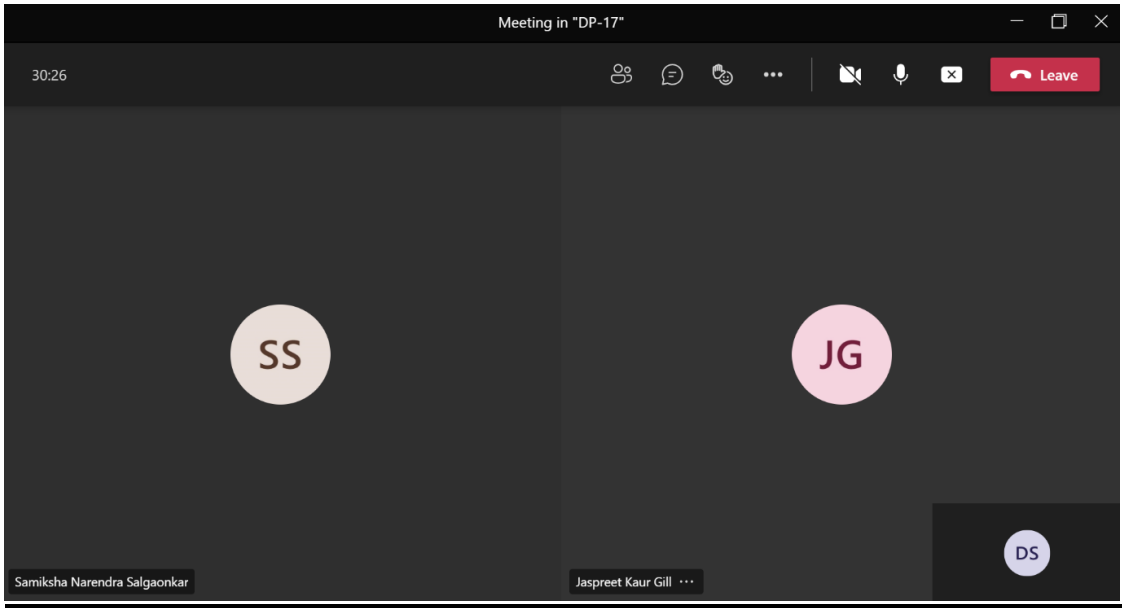

Figure 2: Meeting 1 - "Introduction | Meet & Greet"

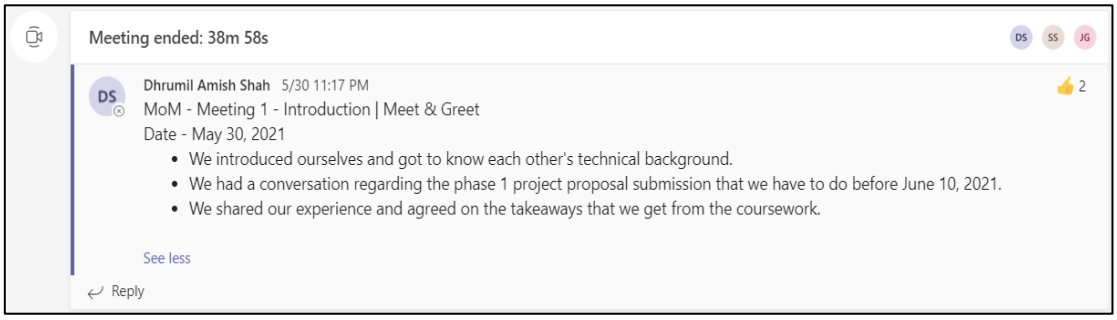

Figure 3 Meeting 1 - MOM

Meeting 2 – Discussion on Project Requirements

Figure 4 and 5 displays the meeting logs and Minutes of the meeting (MoM) of our second meeting respectively.

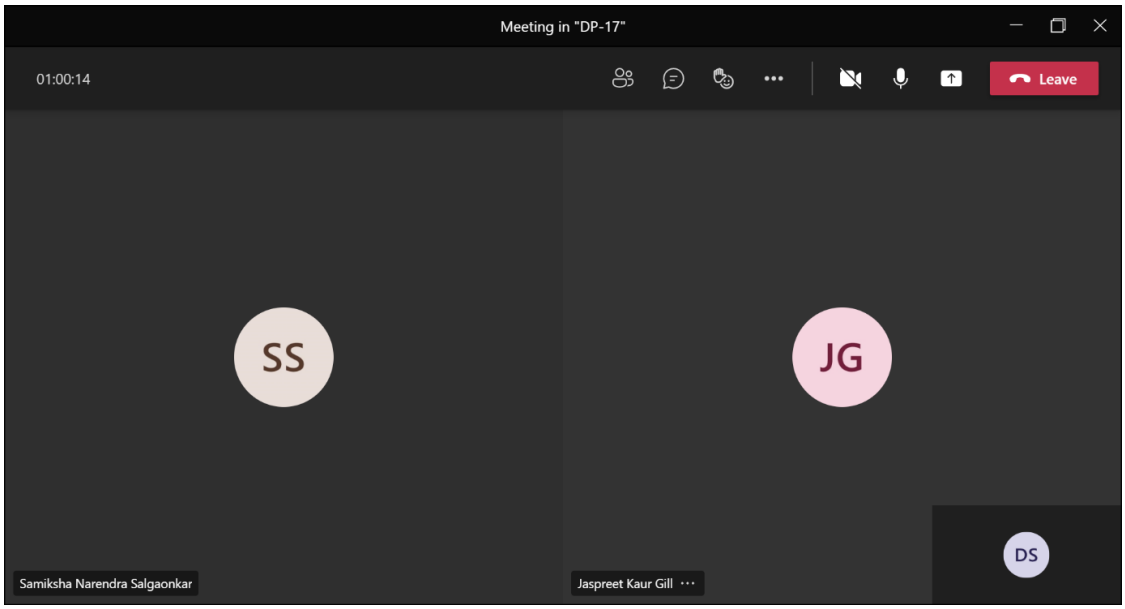

Figure 4: Discussion on Project Requirements

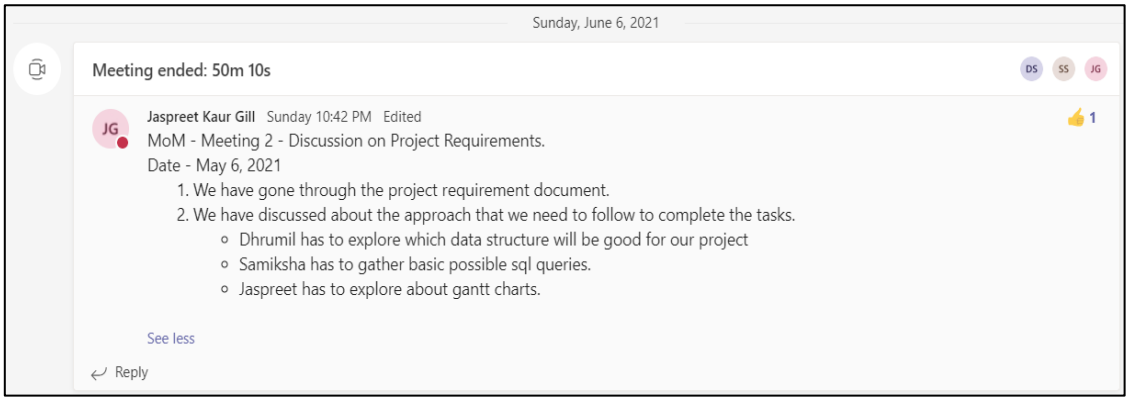

Figure 5: Meeting 2 - MoM

Meeting 3 – Feasibility Analysis and Report Progress Check
Figure 6 displays the Minutes of the meeting (MoM) of our third meeting.

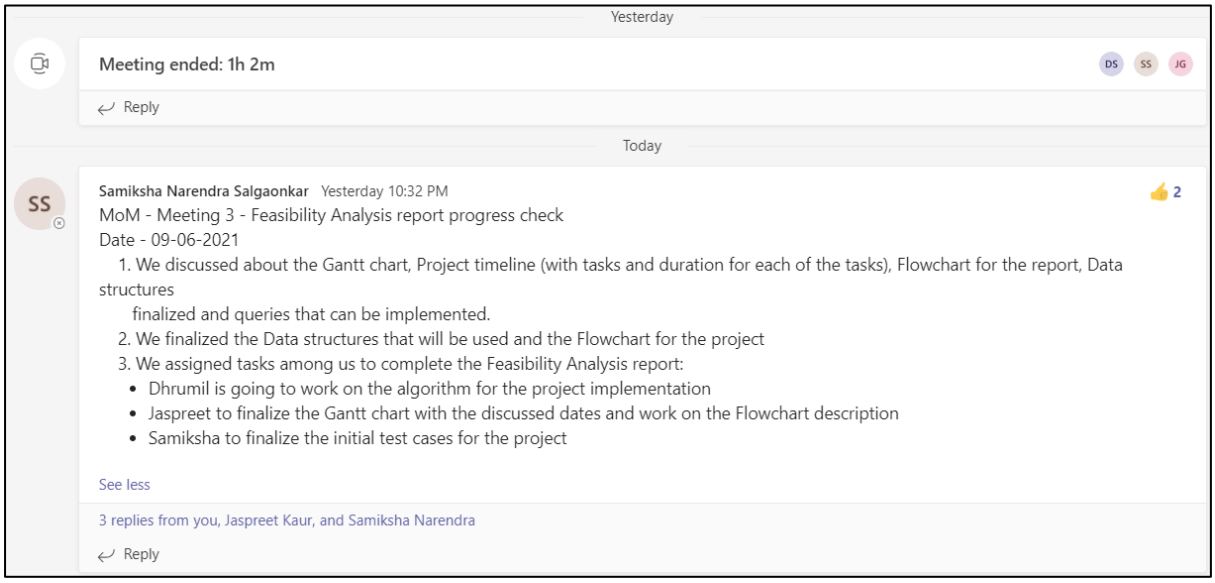

Figure 6: Meeting 3 – MoM

Meeting 4 – Final Report Meeting
Figure 7 displays the meeting logs of our fourth meeting.

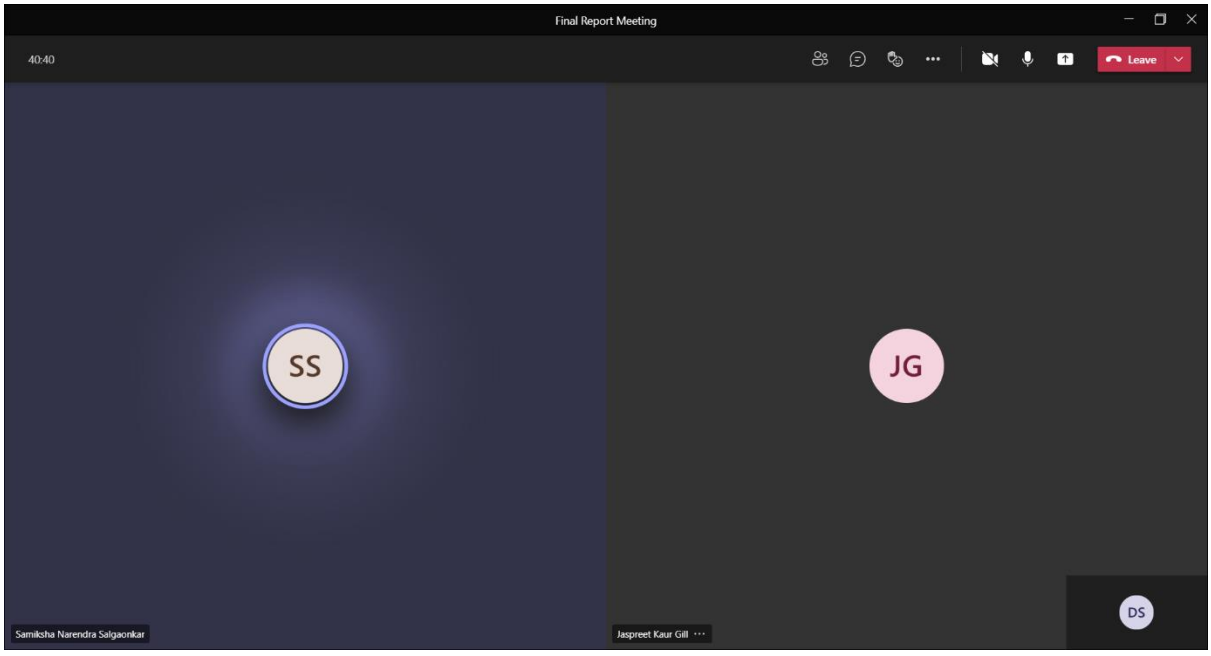

Figure 7: Final document edit

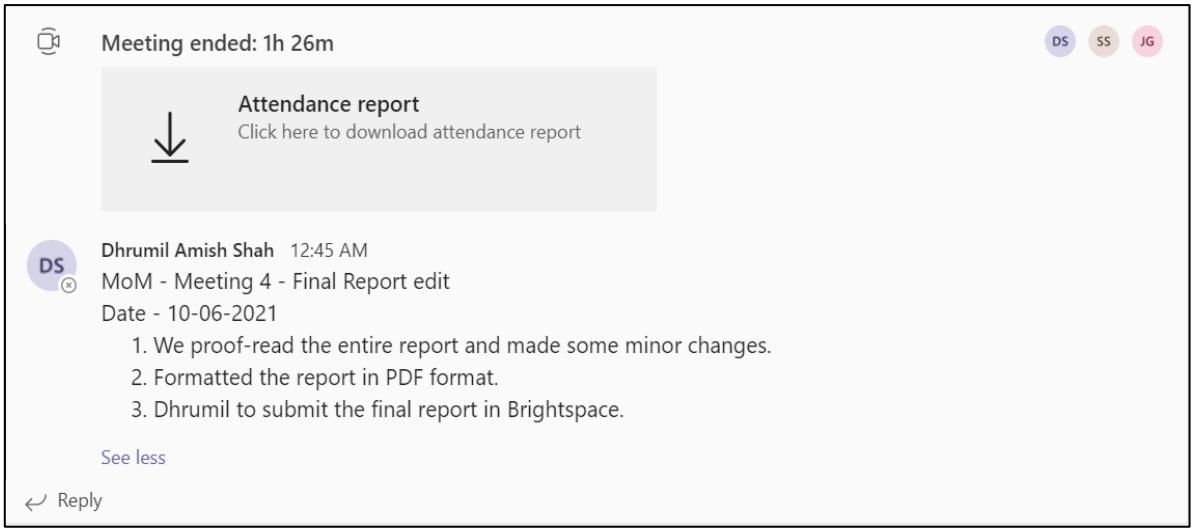

Figure 8: Meeting 4 - MoM