Change request log

1. Concept Location

Step #	Description	Rationale
1	*We ran the system*	
2	*We interacted with the system: after logging in we entered the schedule screen.*	*To get familiar with some of the features of the system, and identify the screens or graphical elements we had to change.*
3	*We inspected the scripts.sql and changed the limit of the password field from varchar(30) to varchar(100)*	*we thought the problem was due to hashed passwords but it did not work*
4	*We used the grep command on the terminal with "password" as the keyword*	*Since the issue was with the password change functionality we wanted to find the file and class responsible for this issue*
5	*From the results appeared we inspected one by the files and found the UsersDao.java file where it was*	*We selected this class because it contained the SQL query for updating user details, including the password.*
6	*We inspected the UserDao.java file and identified the updateUser method.*	*This method was responsible for updating user details, including the password, and was the source of the SQLDataException..*
7	*We marked the UserDao class as "located".*	*We confirmed this class had to be modified to fix the issue.*

Time spent (in minutes): 45

Classes and methods inspected:

- ***UserDao.java***
  - *void updateUser(User user)*
  - *void saveUser(final User user)*
  - *void insertUser(User user)*

2. Impact Analysis

.

Step #	Description	Rationale
1	*We made a list of methods called by updateUser in the UserDao class.*	*To track the classes that could be impacted by the change.*
2	*We inspected the User class.*	*This class is used by updateUser to pass user details, so it might need modifications.*
3	*We inspected the BaseDao class.*	*This is the parent class of UserDao, but we determined it did not need changes.*
4	*We discarded the BaseDao class from the list of classes to change.*	*The issue was specific to the updateUser method in UserDao, and BaseDao was not directly involved.*

Time spent (in minutes): 20

Classes and Methods Inspected:

- ***User.java***
  - *String getUsername()*
  - *String getPassword()*

    - *String getEmail()*
    - *String getPhone()*
- ***BaseDao.java***
    - *void deleteInChunks(String sql, List<Integer> ids)*
    - *String generateUniqueXid(String prefix, String tableName)*

3. Actualization

Step #	Description	Rationale
1	*We modified the updateUser method in UserDao.java to handle null values.*	*To fix the SQLDataException by ensuring VARCHAR fields are not null.*
2	*We replaced null values with empty strings ("") for username, password, email, phone, and homeUrl.*	*This ensures the database does not receive null values for VARCHAR fields.*
3	*We tested the changes by updating user details with null values.*	*To verify that the changes resolved the SQLDataException.*

Time spent (in minutes): 30

Classes and Methods Changed:

- ***UserDao.java***
    - *void updateUser(User user)*

4. Validation

Step #	Description	Rationale
1	*We tested updating user details with null values for username, password, email, phone, and homeUrl.*	*To ensure the changes resolved the SQLDataException.*
2	*We verified that the database stored empty strings instead of null values.*	*To confirm the database behavior matched the expected changes*
3	*We ran the system and interacted with the user management screen.*	*To ensure the system behaved as expected after the changes.*

Time spent (in minutes): 10

5. Summary of the change request

Phase	Time (minutes)	No. of classes inspected	No. of classes changed	No. of methods inspected	No. of methods changes

Concept location	45	1	1	3	1
Impact Analysis	20	2	0	4	0
Prefactoring	0	0	0	0	0
Actualization	30	1	1	1	1
Postfactoring	0	0	0	0	0
Verification	10	1	1	1	1
Total	105	5	3	9	3

6. Conclusions

For this change request, identifying the relevant code (concept location) was pretty straightforward. The issue was clearly tied to the password update functionality, and the `UserDao` class was easy to pinpoint. The main challenge was making sure all `VARCHAR` fields were handled correctly to prevent any `SQLDataException` errors. Since the issue was confined to the `updateUser` method, the impact analysis was minimal. Testing was done manually by interacting with the system and checking the database behavior to confirm everything worked as expected. Overall, the process went smoothly, and the issue was resolved effectively.