

House Rental System Group 5 Final Report

Course/Term:CS5200 Fall 2024

Members:Chunzhang Liu, Haomiao Shi,Ran Cao,Tianmeng Xia

Instructor's Name:Tehmina Amjad

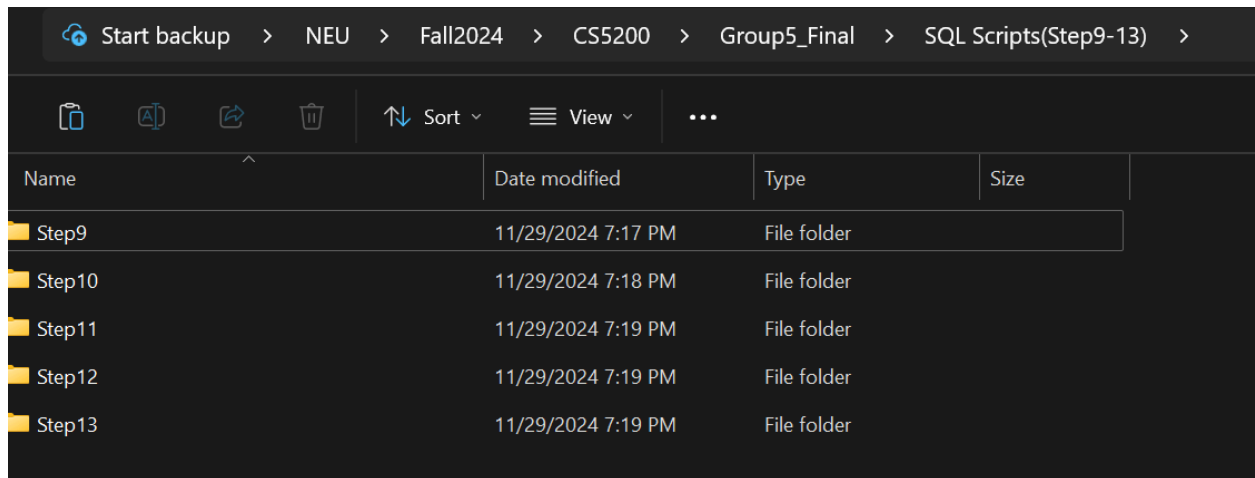
Institution Name: Northeastern University

2. Updated Documentation: Revise Phase I & II deliverables if needed.

2.1 Updated ER Diagram



3. SQL Scripts: All scripts from Steps 9-13. (Zip File)



Step9:

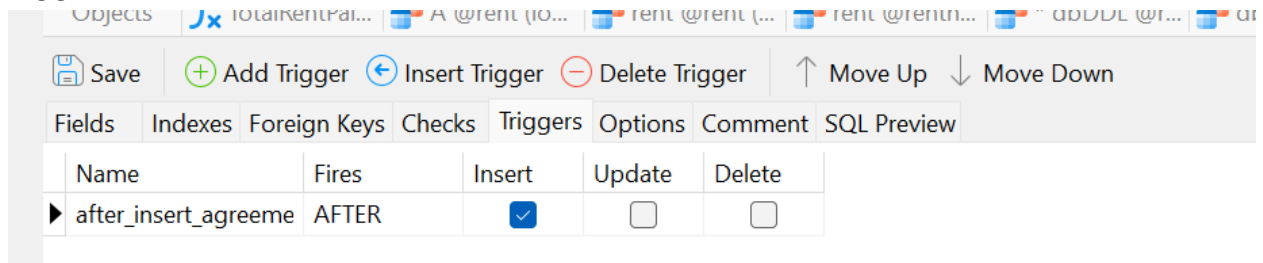
dao from DAO folder in the backend code

Step10:

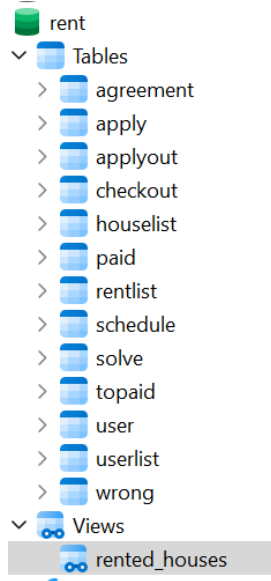
Some updates on the table naming, but the framework is not changed.

We created triggers, view, and functions

trigger:

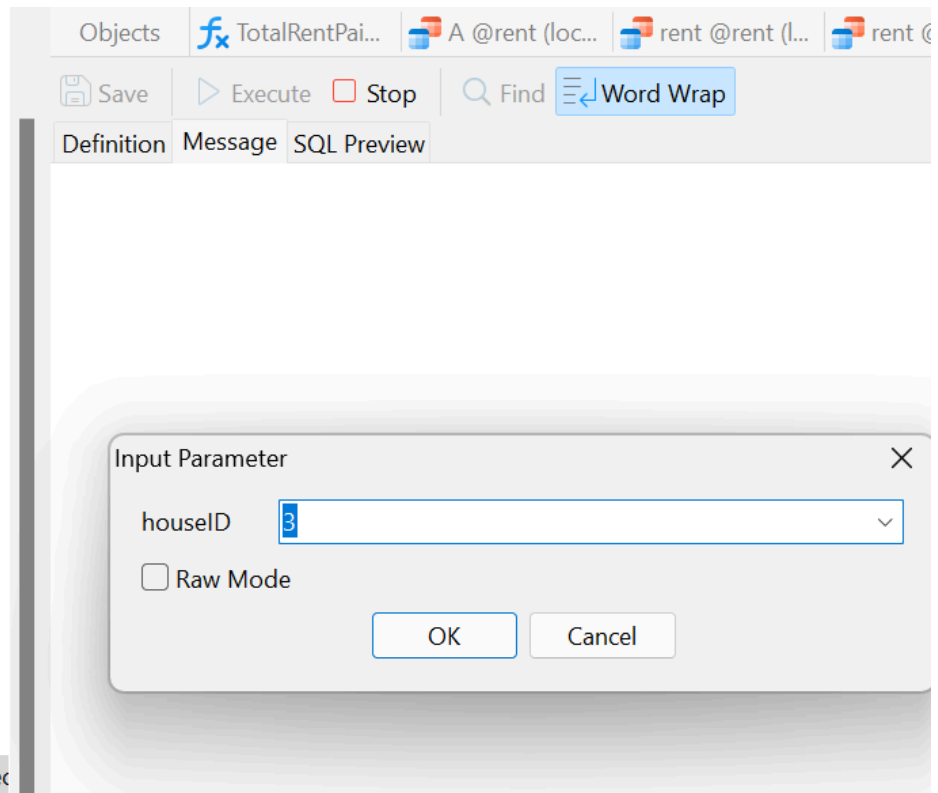
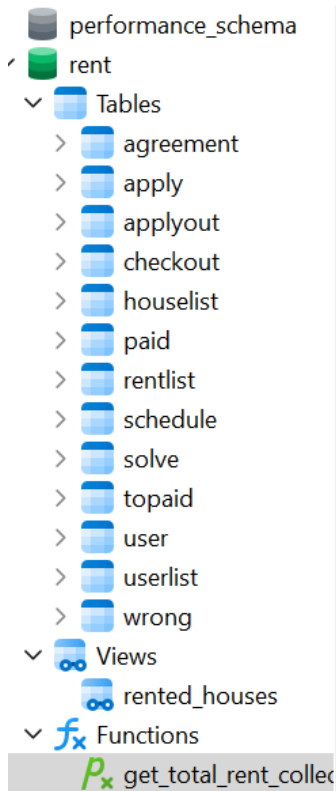


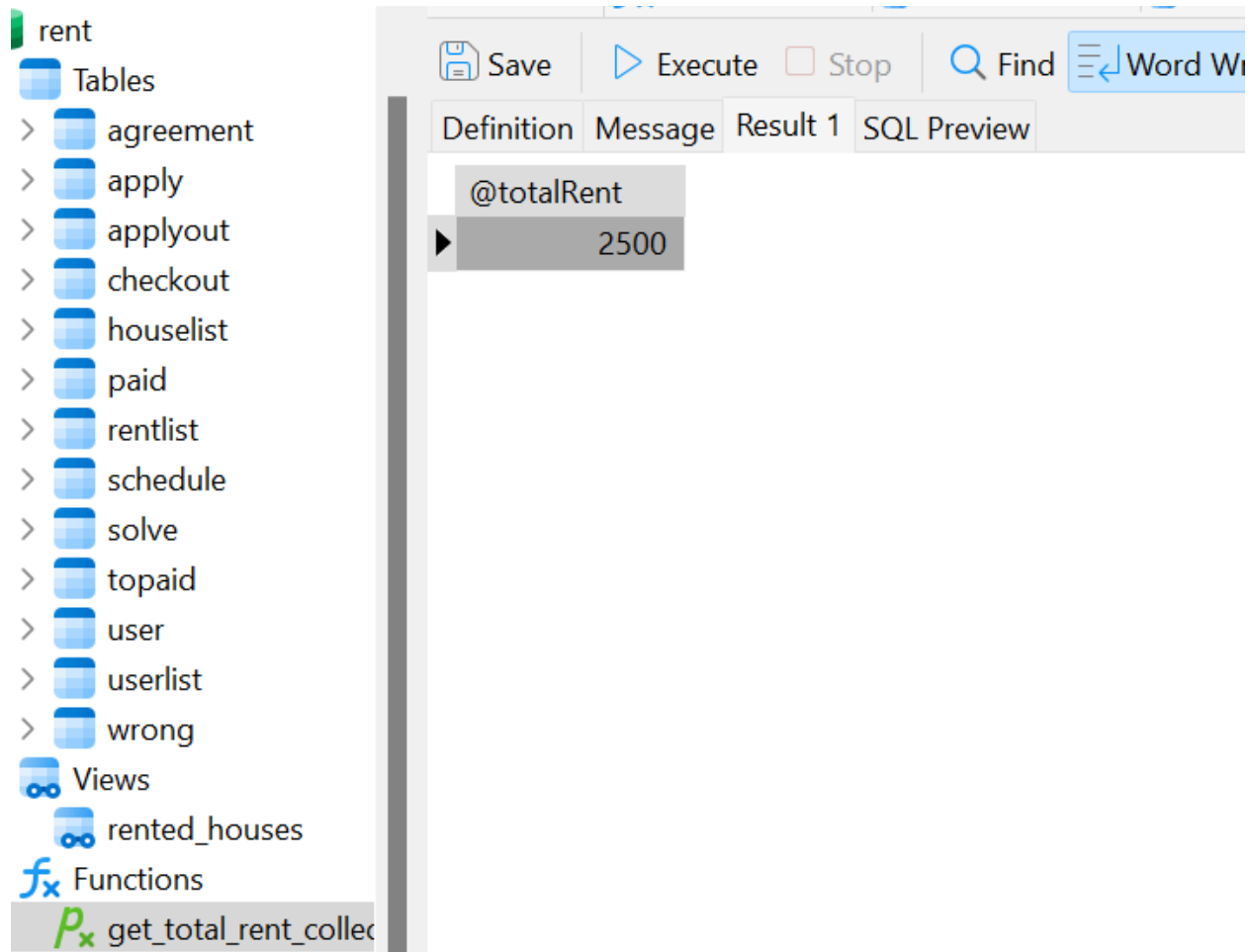
view:



houseid	address	area	price	tenant_name
1	123 Main Street, Apt 301	20	1000	John Doe
3	456 Oak Street, Apt 201	45	2500	Jane Smith
5	789 Pine Avenue, Apt 201	30	1200	Michael Brown
7	202 Elm Street, Apt 102	25	1400	David Wilson
4	101 Maple Street, Apt 501	40	2300	Laura Miller

function:





Step11:

Some updates on the table naming, but the framework is not changed.

Step12:

Some updates on the table naming, but the framework is not changed.

Step13: We created new DQLs to replace origins.

Query1:

- apply
- applyout
- checkout
- houcelist
- paid
- rentlist
- schedule
- solve
- topaid
- user
- userlist
- wrong
- Views
 - rented_houses
- Functions
 - get_total_rent_collec
- Queries
 - dbDDL
 - dbDML
 - dbDROP
 - dbSQL
 - rent
- Backups
 - sys
 - test1

Google Cloud Project1
localhost
atguigudb
customers_info

```

1 -- Query 1: Join query
2 -- 1. Join query: List all rented houses with tenant details
3 SELECT
4   h.houseid AS House_ID,
5   h.address AS Address,
6   h.price AS Rent,
7   u.name AS Tenant_Name,
8   u.phone AS Tenant_Contact
9 FROM houselist h
10 JOIN rentlist r ON h.houseid = r.house_id
11 JOIN userlist u ON r.userlist_id = u.id
12 WHERE h.status = 'Rented';
13
14
15
16 -- Query 2: Aggregate query with GROUP BY and HAVING
17 -- Find the total rent paid by each user, grouped by user, where the total exceeds $1000, sorted by total rent in descending order.
18 SELECT
19   p.userlist_id,
20   u.name AS user_name,
21   SUM(p.price) AS total_rent
22 FROM
23   paid p
24 JOIN
25   userlist u

```

House_ID	Address	Rent	Tenant_Name	Tenant_Contact
1	123 Main Street, Apt 301	1000	John Doe	12165434090
3	456 Oak Street, Apt 201	2500	Jane Smith	12165434091
5	789 Pine Avenue, Apt 201	1200	Michael Brown	12165434093
7	202 Elm Street, Apt 102	1400	David Wilson	12165434095
4	101 Maple Street, Apt 501	2300	Laura Miller	12165434096

Query2:

- agreement
- apply
- applyout
- checkout
- houcelist
- paid
- rentlist
- schedule
- solve
- topaid
- user
- userlist
- wrong
- Views
 - rented_houses
- Functions
 - get_total_rent_collec
- Queries
 - dbDDL
 - dbDML
 - dbDROP
 - dbSQL
 - rent
- Backups
 - sys
 - test1

Google Cloud Project1
localhost
atguigudb
customers_info
contact2

Google Cloud db-gro rent Run Selected Stop Explain Selected

```

16 -- Query 2: Aggregate query with GROUP BY and HAVING
17 -- Find the total rent paid by each user, grouped by user, where the total exceeds $1000, sorted by total rent in descending order.
18 SELECT
19   p.userlist_id,
20   u.name AS user_name,
21   SUM(p.price) AS total_rent
22 FROM
23   paid p
24 JOIN
25   userlist u
26 ON
27   p.userlist_id = u.id
28 GROUP BY
29   p.userlist_id, u.name
30 HAVING
31   total_rent > 1000
32 ORDER BY
33   total_rent DESC;
34
35 -- Query 3: Subquery
36 -- Find all houses that have been applied for but not yet added to the 'applyout' table.
37 SELECT
38   houseid, address
39 FROM
40   houselist

```

userlist_id	user_name	total_rent
3	Jane Smith	2500
5	Michael Brown	2400
2	John Doe	2300
7	David Wilson	1400
4	Emily Johnson	1100

Query3:

user

userlist

wrong

Views

rented_houses

Functions

get_total_rent_collec

Queries

dbDDL

dbDML

dbDROP

dbSQL

rent

Backups

;

it1

gle Cloud Project1

33total_rent DESC;

34

35-- Query 3: Subquery

36-- Find all houses that have been applied for but not yet added to the `applyout` table.

37SELECT

38houseid, address

39FROM

40houselist

41WHERE

42houseid NOT IN (

43SELECT house_id FROM applyout

44);

45

46

MessageSummaryResult 1Result 2Result 3Result 4Result 5ProfileStatus

houseid	address
7	202 Elm Street, Apt 102

Query4:

applyout

checkout

houselist

paid

rentlist

schedule

solve

topaid

user

userlist

wrong

Views

rented_houses

Functions

get_total_rent_collec

Queries

dbDDL

dbDML

dbDROP

dbSQL

rent

Backups

sys

test1

Google Cloud Project1

localhost

atguigudb

customers_info

dbtest2

dbtest12

dbtest14

46

47-- Query 4: Join with multiple tables

48-- Find all tenants and the corresponding contracts with their rental house details.

49SELECT

50a.tenant AS tenant_name,

51a.address AS agreement_address,

52h.area AS house_area,

53h.price AS house_price,

54a.fromdate,

55a.todate

56FROM

57agreement a

58JOIN

59houselist h

60ON

61a.house_id = h.houseid;

62

63-- Query 5: Aggregate query with GROUP BY, HAVING, and inline subquery

64-- Find the average rental price for each user and filter users who rented houses larger than the average house area.

65SELECT

66p.userlist_id,

67u.name AS user_name,

68AVG(p.price) AS avg_rent

69FROM

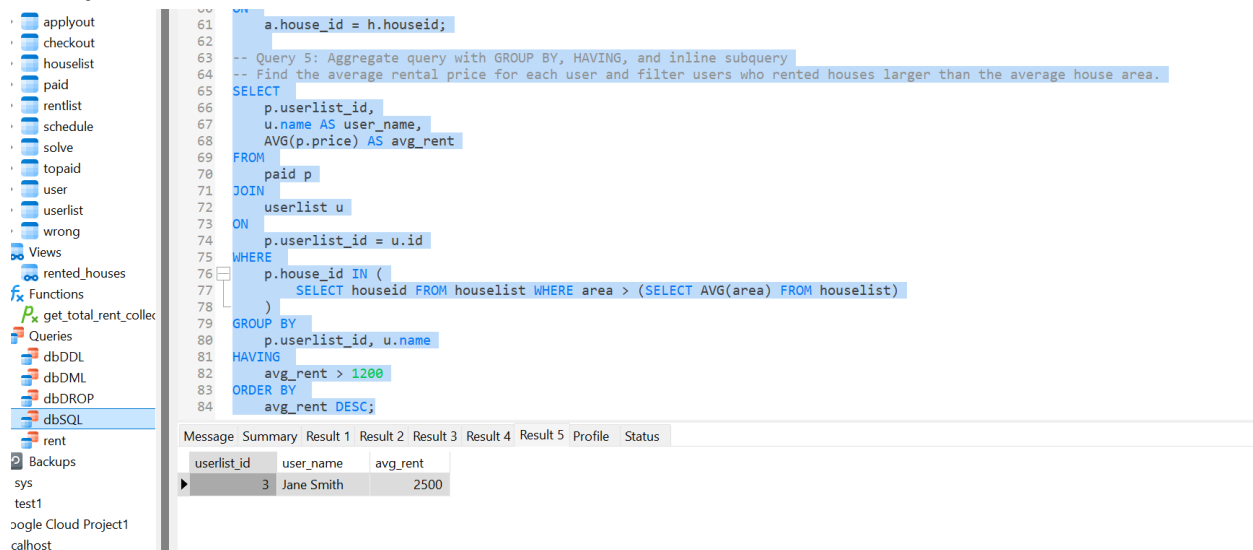
70paid p

71

MessageSummaryResult 1Result 2Result 3Result 4Result 5ProfileStatus

tenant_name	agreement_address	house_area	house_price	fromdate	todate
John Doe	123 Main Street, Apt 301	20	1000	2024-01-04	2025-01-01
Emily Johnson	456 Oak Street, Apt 402	22	1100	2024-02-01	2025-02-01
Jane Smith	456 Oak Street, Apt 201	45	2500	2024-01-04	2024-10-03
Laura Miller	101 Maple Street, Apt 501	40	2300	2024-06-01	2025-06-01
Michael Brown	789 Pine Avenue, Apt 201	30	1200	2024-03-01	2025-03-01
Sarah Davis	101 Maple Street, Apt 501	35	1300	2024-04-01	2025-04-01
David Wilson	202 Elm Street, Apt 102	25	1400	2024-05-01	2025-05-01

Query5:



The screenshot shows a database IDE with a project explorer on the left and a SQL editor on the right. The project explorer lists various tables and views, including 'rented_houses', 'Functions', 'Queries', 'dbDDL', 'dbDML', 'dbDROP', 'dbSQL', 'rent', 'Backups', 'sys', 'test1', 'Google Cloud Project1', and 'calhost'. The SQL editor contains the following query:

```

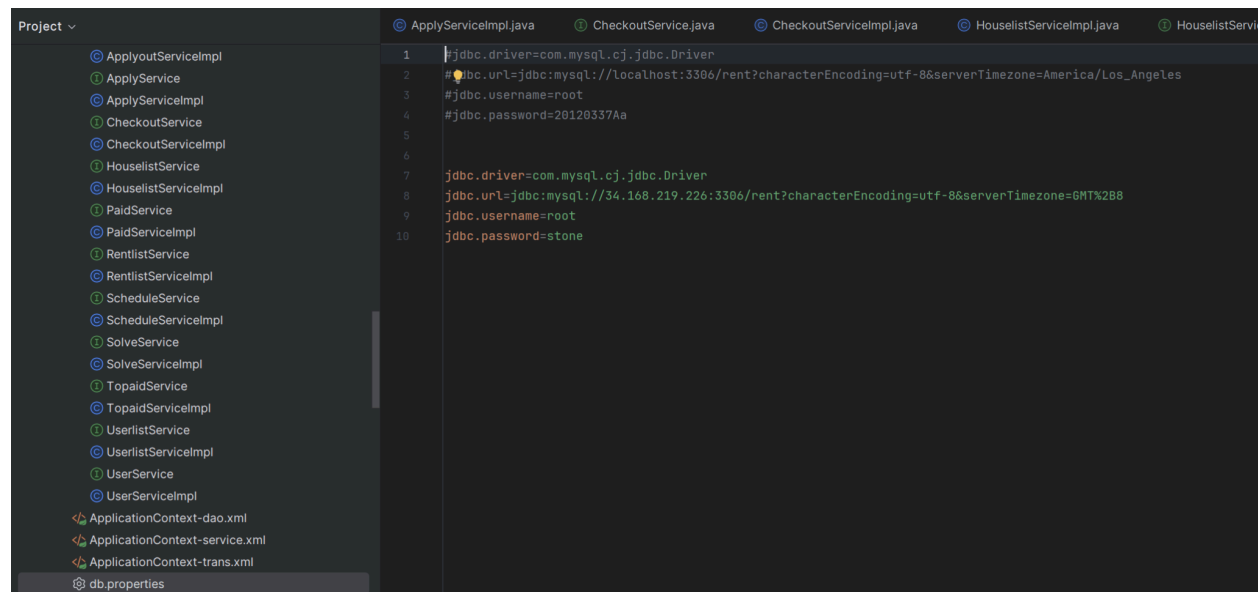
61 a.house_id = h.houseid;
62
63 -- Query 5: Aggregate query with GROUP BY, HAVING, and inline subquery
64 -- Find the average rental price for each user and filter users who rented houses larger than the average house area.
65 SELECT
66     p.userlist_id,
67     u.name AS user_name,
68     AVG(p.price) AS avg_rent
69 FROM
70     paid p
71 JOIN
72     userlist u
73 ON
74     p.userlist_id = u.id
75 WHERE
76     p.house_id IN (
77         SELECT houseid FROM housedlist WHERE area > (SELECT AVG(area) FROM housedlist)
78     )
79 GROUP BY
80     p.userlist_id, u.name
81 HAVING
82     avg_rent > 1200
83 ORDER BY
84     avg_rent DESC;

```

Below the query, a message bar shows the results of the query. The results are displayed in a table with the following columns: userlist_id, user_name, and avg_rent. The results are as follows:

userlist_id	user_name	avg_rent
3	Jane Smith	2500

Attention: Below is for GCP, Above is for Local



The screenshot shows an IDE with a project explorer on the left and a Java source code editor on the right. The project explorer lists various Java classes and XML files, including 'ApplyoutServiceImpl', 'ApplyService', 'ApplyServiceImpl', 'CheckoutService', 'CheckoutServiceImpl', 'HouselistService', 'HouselistServiceImpl', 'PaidService', 'PaidServiceImpl', 'RentlistService', 'RentlistServiceImpl', 'ScheduleService', 'ScheduleServiceImpl', 'SolveService', 'SolveServiceImpl', 'TopaidService', 'TopaidServiceImpl', 'UserlistService', 'UserlistServiceImpl', 'UserService', 'UserServiceImpl', 'ApplicationContext-dao.xml', 'ApplicationContext-service.xml', 'ApplicationContext-trans.xml', and 'db.properties'. The Java source code editor shows the following code:

```

1 jdbc.driver=com.mysql.cj.jdbc.Driver
2 #jdbc.url=jdbc:mysql://localhost:3306/rent?characterEncoding=utf-8&serverTimezone=America/Los_Angeles
3 #jdbc.username=root
4 #jdbc.password=20120337Aa
5
6
7 jdbc.driver=com.mysql.cj.jdbc.Driver
8 jdbc.url=jdbc:mysql://34.168.219.226:3306/rent?characterEncoding=utf-8&serverTimezone=GMT%288
9 jdbc.username=root
10 jdbc.password=stone

```

4. Application Source Code: Include frontend and backend code with setup instructions. (GitHub Links)

<https://github.com/stone-coding/CS5200Group5>

5. Conclusion: ○ Reflect on challenges, learnings, and improvements.

challenges:

- Database Design Complexity: Ensuring proper normalization while accommodating all functional requirements was a significant challenge. Balancing between avoiding redundancy and maintaining performance was a key consideration.
 - Integration Issues: Configuring the backend to work seamlessly with the MySQL database on GCP required troubleshooting connectivity issues and optimizing queries for remote access.
 - Session Management: Handling session conflicts between admin and tenant users on the same browser exposed flaws in our session management logic, requiring additional debugging and testing.
 - Technology Learning Curve: For some team members, working with SpringMVC and MyBatis was a new experience, which initially slowed down the development process.
-

Learnings

Despite the challenges, the project provided invaluable learning opportunities:

- Database Optimization: We gained hands-on experience in designing scalable databases, indexing for performance, and using SQL scripts for efficient data population.
 - Framework Proficiency: Working with Spring, SpringMVC, and MyBatis deepened our understanding of backend frameworks and best practices for building maintainable codebases.
 - Cloud Integration: Setting up and connecting to a MySQL instance on GCP improved our skills in cloud database management and taught us the importance of secure configurations.
 - Collaborative Development: The project emphasized the importance of clear communication, version control, and task delegation in team environments.
-

Improvements

Looking forward, we identified areas for improvement that could elevate the quality of similar projects in the future:

- Error Handling: Implementing more robust error-handling mechanisms, especially for session conflicts and database connectivity issues, would improve the reliability of the application.
- Testing Strategy: Adopting a more structured testing approach, such as automated unit and integration tests, could ensure higher code quality and catch issues earlier in development.
- Frontend Optimization: The user experience on the frontend can be enhanced by introducing responsive design techniques and improving the interface for better usability.
- Scalability Considerations: Refactoring parts of the application to support horizontal scaling and larger datasets would prepare the system for real-world deployment scenarios.