

Spring Boot
XXX XXXXXXXX
XXX XXX
2026 X X

“ ”
“ + ”
+ Vite + Element Plus Spring Boot MyBatis-Plus JWT MySQL Vue 3

Spring Boot MySQL

Abstract

This report presents the design and implementation of a homestay recommendation system. The system targets the information overload in short-term rental platforms and uses a hybrid recommendation strategy combining collaborative filtering and content similarity. The frontend is implemented with Vue 3 and Element Plus, while the backend uses Spring Boot, MyBatis-Plus, JWT authentication, and MySQL storage. The report follows the database course project template, covering requirement analysis, conceptual and logical database design, physical implementation, and database operations, and further describes the overall architecture and recommender logic. The project provides a practical reference for similar personalized recommendation systems.

Keywords: Homestay recommendation; Collaborative filtering; Content-based similarity; Spring Boot; MySQL

	2022105420004 XXX

	1		2
	E-R	3	
	4	MySQL	
	5		

1. \
2. \
3. \
4. \
5. \
6. \
7. \
8. \
9. \
- 10.

1

1.1

“ + ”
“ + ”

“ ”
“ ”

1.2

-
-
-
-
-
-

1.3

- 1
- 2

1.4

-
-
-
-
-
-

JWT

1.5

-
-
-
-
-

JWT

1.6

1.7

-
-
-
-

Token

1.8

1.9

DAU/MAU

1.10

-

A/B

1.11

A/B

1.12

A/B

1.13

A/B

1.14

A/B

1.15

A/B

1.16

A/B

1.17

A/B

1.18

Precision	Recall	NDCG	A/B
		A/B	

1.19

A/B

1.20

1.21

2.1

Recommendation		User	Property	Order
PropertyImage	City	Interaction	Facility	
		Tag		

2.2 **E-R**

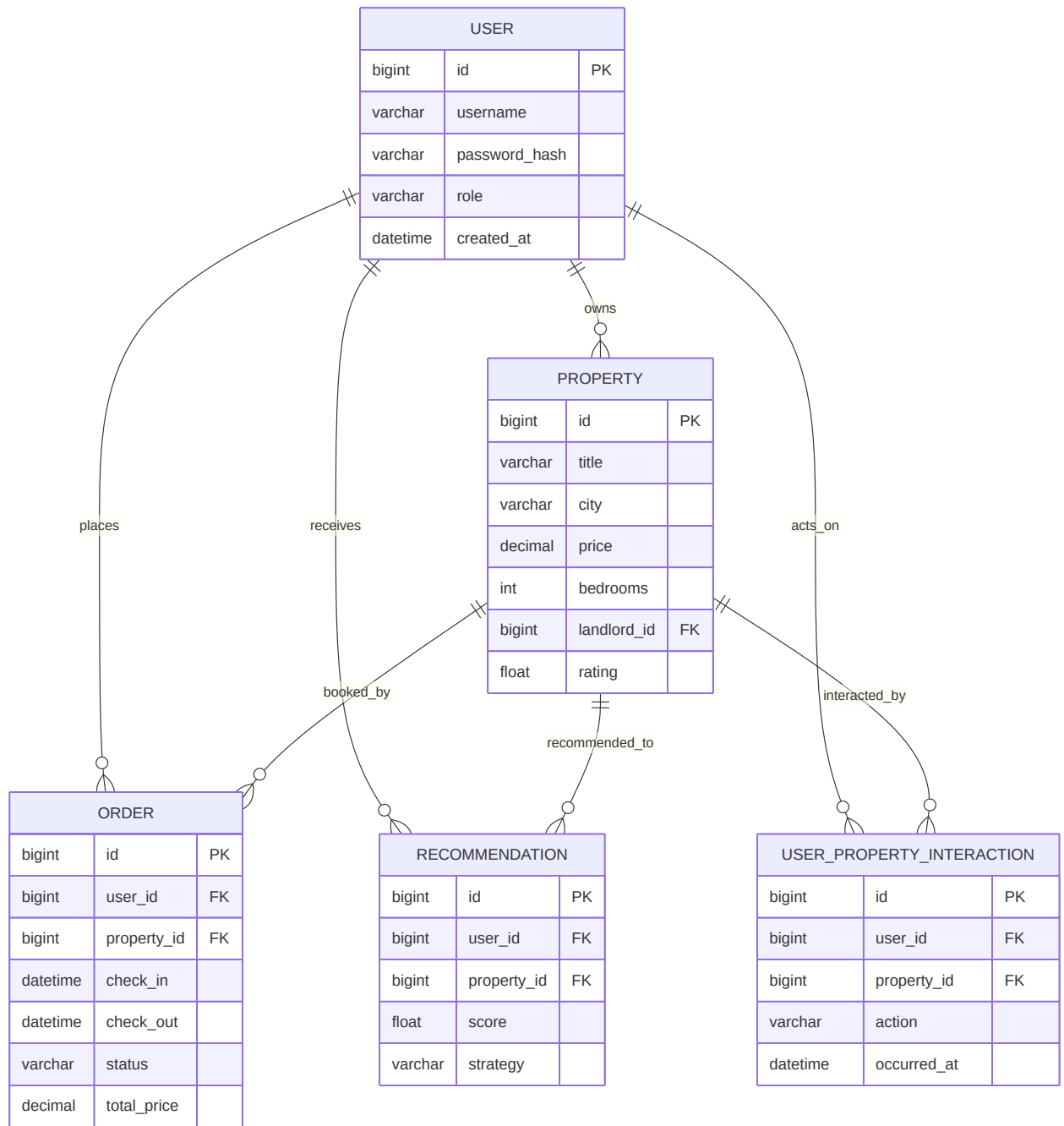
E-R

-
-
-
-
-

2.3 **E-R**

- -

2.4 **E-R**



E-R

-

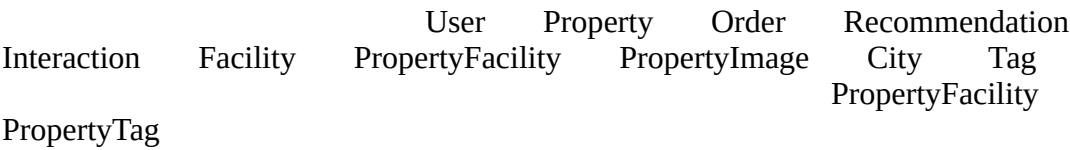
2.5

“ - - ”

AB

3

3.1 E-R



3.2

3.3

user		
id	BIGINT	
username	VARCHAR(50)	
password_hash	VARCHAR(255)	
phone	VARCHAR(20)	
email	VARCHAR(100)	
role	VARCHAR(20)	USER/LANDLORD/ADMIN
status	VARCHAR(20)	
created_at	DATETIME	
updated_at	DATETIME	

property		
id	BIGINT	
title	VARCHAR(200)	
city	VARCHAR(50)	
address	VARCHAR(200)	

price	DECIMAL(10,2)	
bedrooms	INT	
max_guests	INT	
rating	DECIMAL(3,2)	
landlord_id	BIGINT	ID
status	VARCHAR(20)	
created_at	DATETIME	

order

id	BIGINT	
user_id	BIGINT	
property_id	BIGINT	
check_in	DATE	
check_out	DATE	
total_price	DECIMAL(10,2)	
status	VARCHAR(20)	
created_at	DATETIME	

interaction

id	BIGINT	
user_id	BIGINT	
property_id	BIGINT	
action	VARCHAR(20)	VIEW/FAVOR/BOOK
weight	INT	
occurred_at	DATETIME	

recommendation

id	BIGINT	

user_id	BIGINT	
property_id	BIGINT	
score	DECIMAL(6,4)	
strategy	VARCHAR(50)	
created_at	DATETIME	

facility

id	BIGINT	
name	VARCHAR(50)	
category	VARCHAR(50)	
description	VARCHAR(200)	

property_facility

id	BIGINT	
property_id	BIGINT	ID
facility_id	BIGINT	ID
created_at	DATETIME	

property_image

id	BIGINT	
property_id	BIGINT	ID
url	VARCHAR(255)	
sort_order	INT	
created_at	DATETIME	

3.4

PENDING/CONFIRMED/CANCELLED/COMPLETED

3.5

(user_id, created_at)
(property_id, occurred_at)
user.username user.phone

4

4.1

MySQL

```
CREATE DATABASE homestay_recommendation DEFAULT CHARACTER SET utf8mb4
```

utf8mb4

4.2

```
CREATE TABLE property (  
  id BIGINT PRIMARY KEY AUTO_INCREMENT,  
  title VARCHAR(200) NOT NULL,  
  city VARCHAR(50) NOT NULL,  
  address VARCHAR(200) NOT NULL,  
  price DECIMAL(10,2) NOT NULL,  
  bedrooms INT NOT NULL,  
  max_guests INT NOT NULL,  
  landlord_id BIGINT NOT NULL,  
  rating DECIMAL(3,2) DEFAULT 0,  
  status VARCHAR(20) DEFAULT 'ACTIVE',  
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```

4.3

```
CREATE VIEW view_hot_property AS
SELECT property_id, COUNT(*) AS order_count
FROM `order`
GROUP BY property_id;
```

4.4

city price rating

```
CREATE INDEX idx_property_city ON property(city);
CREATE INDEX idx_property_price ON property(price);
CREATE INDEX idx_property_rating ON property(rating);
```

user_id property_id
(property_id, check_in)

4.5

```
CREATE TRIGGER trg_update_rating AFTER INSERT ON review
FOR EACH ROW
BEGIN
    UPDATE property
    SET rating = (SELECT AVG(score) FROM review WHERE property_id
    WHERE id = NEW.property_id;
END;
```

4.6

```
CREATE PROCEDURE sp_get_recommendations(IN uid BIGINT)
BEGIN
    SELECT property_id, score
    FROM recommendation
    WHERE user_id = uid
    ORDER BY score DESC
    LIMIT 20;
END;
```

4.7

InnoDB

4.8

5

5.1

1

```
SELECT * FROM property
WHERE city = '□□'
ORDER BY rating DESC
LIMIT 10;
```

2

```
SELECT * FROM `order`
WHERE user_id = 1001
ORDER BY created_at DESC;
```

3

```
SELECT * FROM interaction
WHERE property_id = 2001 AND action = 'VIEW'
ORDER BY occurred_at DESC;
```

4

```
SELECT p.*, r.score
FROM recommendation r
```

```
JOIN property p ON r.property_id = p.id
WHERE r.user_id = 1001
ORDER BY r.score DESC
LIMIT 20;
```

5.2

-

```
INSERT INTO `order` (user_id, property_id, check_in, check_out,
VALUES (1001, 2001, '2026-03-01', '2026-03-05', 1200.00, 'PENDIN
```

-

```
UPDATE property
SET price = 499.00
WHERE id = 2001;
```

-

```
INSERT INTO interaction (user_id, property_id, action, occurred_
VALUES (1001, 2001, 'BOOK', NOW());
```

-

```
UPDATE `order`
SET status = 'CANCELLED'
WHERE id = 3001;
```

5.3

```
START TRANSACTION;
UPDATE property_calendar
SET available = 0
WHERE property_id = 2001 AND date BETWEEN '2026-03-01' AND '2026
INSERT INTO `order` (user_id, property_id, check_in, check_out,
VALUES (1001, 2001, '2026-03-01', '2026-03-05', 1200.00, 'PENDIN
COMMIT;
```

5.4

is_deleted

6

6.1

6.2

-
-
-
-
-
-

SQL

6.3

6.4

7

7.1

RESTful API
Vue 3 Element Plus MyBatis-Plus

MySQL Spring Boot

7.2

- JWT Axios Token
-
-
-
-

7.3

-

3 recommendation 1 2

7.4 API

API
/api/auth /api/property /api/order /api/recommendation

7.5

Maven mvn spring-boot:run Vite npm run
build MySQL 8.x Redis
Linux + Docker JWT ELK Prometheus
+ Grafana

7.6

Element Plus

7.7

JWT

ELK

8

8.1

8.2

Postman
401

Rest Client
403

API
400

500

Token

8.3

JMeter

SQL

95%

8.4

8.5

TC-01			Token
TC-02		= <500	
TC-03			
TC-04			
TC-05			401

9

10

- [1] Resnick P, Varian H R. Recommender systems. Communications of the ACM, 1997.
 - [2] Sarwar B, et al. Item-based Collaborative Filtering Recommendation Algorithms. WWW, 2001.
 - [3] He X, et al. Neural Collaborative Filtering. WWW, 2017.
 - [4] . . , 2016.
 - [5] . . , 2019.
-

11

11.1

			200+
1000+	300+		

11.2

11.3

```
npm run build                                mvn spring-boot:run
sql/schema.sql  sql/sample_data.sql
```

11.4

			Precision@K
Recall@K	NDCG	A/B	

11.5

11.6

11.7

11.8 A/B

A/B